

AD-A040 049

SPERRY UNIVAC ST PAUL MINN DEFENSE SYSTEMS DIV
MODERN PROGRAMMING PRACTICES STUDY REPORT.(U)

F/G 9/2

APR 77 W E BRANNING, D M WILLSON

F30602-76-C-0136

UNCLASSIFIED

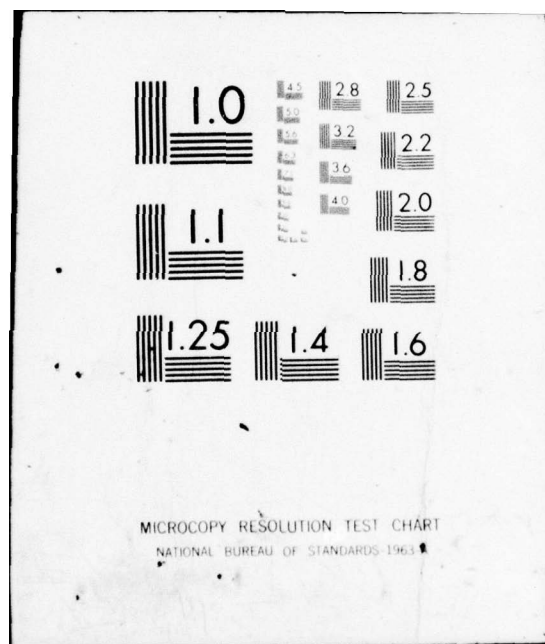
PX-11932-F

RADC-TR-77-106

NL

1 OF 5
AD
AO40049





AD A 040049

RADC-TR-77-106
Final Technical Report
April 1977

MODERN PROGRAMMING PRACTICES STUDY REPORT

Sperry Univac

Approved for public release; distribution unlimited.



AD No. —

DDC FILE COPY

THE AIR FORCE RESEARCH AND DEVELOPMENT COMMAND
ATTENTION: TECHNICAL SERVICES
3901 RANDOLPH ROAD, RANDOLPH AFB, TEXAS 78151

This report contains some machine-produced copy which is not of the highest printing quality, but because of economical considerations, it was determined in the best interest of the government that they be used in this publication.

The notices of proprietary data are only referenced on pages 1-23, 3-16 and 9-17 of the text and do not contain technical details within this document.

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

This report has been reviewed and approved for publication.

APPROVED:

Roger W. Weber
ROGER W. WEBER
Project Engineer

APPROVED:

Alan R. Barnum
ALAN R. BARNUM
Assistant Chief
Information Sciences Division

FOR THE COMMANDER:

John P. Huss
JOHN P. HUSS
Acting Chief, Plans Office

Do not retain copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-77-106	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MODERN PROGRAMMING PRACTICES STUDY REPORT	5. TYPE OF REPORT & PERIOD COVERED Final Technical Report Jan 76 - Jan 77	6. PERFORMING ORG. REPORT NUMBER PX-11932-F
7. AUTHOR(s) W. E. Branning, J. P. Schaezner D. M. Willson, W. A. Erickson	8. CONTRACT OR GRANT NUMBER(s) F30602-76-C-0136	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Sperry Univac/Systems Software Engineering Sperry Univac Defense Systems, Univac Park St Paul MN 55165	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 55810272	
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (ISIM) Griffiss AFB NY 13441	12. REPORT DATE Apr 1977	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same	13. NUMBER OF PAGES 419	
	15. SECURITY CLASS. (of this report) UNCLASSIFIED	
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited. 16 5581 1702		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Same		
18. SUPPLEMENTARY NOTES RADC Project Engineer: Roger W. Weber (ISIM)		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Modern Programming Practices, Software Development, Real-time Command and Control Systems, AN/UYK-7 Computer, Top-Down Program Development, Software tools.		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This technical report and report summary show that for the surveyed program developments: (a) Top-down program development reduced cost by 15%. (b) Regardless of the method for development used labor was generally distributed as follows:		

DD FORM 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

408125

DDC
RECEIVED
JUN 1 1977
C

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

- (1) 10% for analysis,
- (2) 30% for design,
- (3) 35% for coding and debugging,
- (4) 25% for testing,
- c. Program testing was automated using real-time on line-simulation, scenario control, data recording and reduction.
- d. Software quality assurance and configuration management were formal disciplines.
- e. Through system design, operating system software provided real-time on-line casualty recovery of system functions from hardware failures.
- f. Through system design, operating system software allowed for on-line maintenance testing of hardware equipments while continuing real-time operation.
- g. Software tools found effective on the surveyed programs were described and evaluated.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

TABLE OF CONTENTS

<u>Paragraph</u>	<u>Title</u>	<u>Page</u>
	List of Figures	xiv
	List of Tables	xiii
	List of Acronyms	xvi
	Study Report Summary	1
Section 1	Program Environments (Task 1.2)	1-1
1.0	INTRODUCTION	1-1
1.1	Program Descriptions	1-2
1.1.1	Program 1	1-2
1.1.1.1	Operating System	1-2
1.1.1.2	Navigation Program	1-5
1.1.1.3	Navigation Simulation Program	1-6
1.1.1.4	Reduced Capability Program	1-7
1.1.2	Program 2	1-8
1.1.2.1	Operating System	1-8
1.1.2.2	Test Program	1-8
1.1.2.3	Operational Program	1-8
1.1.2.4	Simulation Program	1-9
1.1.3	Program 3	1-12
1.1.4	Program 4	1-12
1.2	Program Baselineing Through System Test/ Evaluation	1-13
1.3	Target Hardware Suites	1-14
1.3.1	Program 1	1-14
1.3.2	Program 2	1-16
1.3.3	Program 3	1-17
1.3.4	Program 4	1-19
1.4	Special Requirements	1-19

TABLE OF CONTENTS (continued)

<u>Paragraph</u>	<u>Title</u>	<u>Page</u>
1.4.1	Program 1	1-19
1.4.2	Program 2	1-20
1.4.3	Program 3	1-20
1.4.4	Program 4	1-21
1.5	Customer Furnished Data	1-21
1.5.1	Program 1	1-21
1.5.2	Program 2	1-22
1.5.3	Program 3	1-22
1.5.4	Program 4	1-23
1.6	Furnished Support Software	1-23
1.7	Local and Remote Data Processing	1-24
1.8	Delivered Items	1-25
1.9	Major Contract Milestones Schedule	1-27
1.10	Manpower Schedules	1-27
1.11	Program Personnel	1-27
Section 2	Top Down Program Development (Task 2.1)	2-1
2.0	INTRODUCTION	2-1
2.1	Methodology for Top Down Program Development	2-2
2.2	Program 3 and 4 Top Down Development	2-5
2.3	Top Down Development Techniques	2-10
2.3.1	Executive Program Constraints	2-12
2.3.2	Program Language Elements	2-14
2.3.3	Top Down Documentation	2-15
2.3.4	Top Down Rules	2-21
2.3.5	Top Down Production	2-22
2.3.6	Top Down Testing	2-25
2.4	Summary	2-27
2.5	Conclusion	2-28

TABLE OF CONTENTS (continued)

<u>Paragraph</u>	<u>Title</u>	<u>Page</u>
Section 3	Software Development Standards (Tasks 2.2)	3-1
3.0	INTRODUCTION	3-1
3.1	Documented Software Development Standards	3-1
3.1.1	Program 1	3-2
3.1.2	Program 2	3-3
3.1.3	Programs 3 and 4	3-5
3.2	Development Methods and Tools	3-9
3.2.1	Program 1	3-10
3.2.2	Program 2	3-10
3.2.2.1	Planning	3-10
3.2.2.2	Analysis	3-12
3.2.2.3	Specification	3-12
3.2.2.4	Design	3-13
3.2.2.5	Code and Debug	3-15
3.2.2.6	Testing	3-17
3.2.2.7	Configuration Management	3-18
3.2.3	Program 3	3-19
3.2.3.1	Planning	3-19
3.2.3.2	Analysis and Specification	3-24
3.2.3.3	Design	3-25
3.2.3.4	Code and Debug	3-26
3.2.3.5	Testing	3-28
3.2.3.6	Configuration Management	3-30
3.2.4	Program 4	3-31
3.2.4.1	Planning	3-31
3.2.4.2	Analysis	3-40
3.2.4.2.1	Inputs to Analysis Tasks	3-41
3.2.4.2.2	Analysis Description	3-44
3.2.4.2.3	Reports	3-47
3.2.4.3	Design	3-49

TABLE OF CONTENTS (continued)

<u>Paragraph</u>	<u>Title</u>	<u>Page</u>
3.2.4.3.1	Design Inputs	3-49
3.2.4.3.2	Design Process	3-50
3.2.4.3.2.1	Program Structure	3-50
3.2.4.3.2.2	Implementation Approach	3-52
3.2.4.3.2.3	Test Requirements	3-53
3.2.4.3.3	Design Outputs	3-54
3.2.4.3.4	Design Reviews	3-54
3.2.4.4	Code and Debug	3-54
3.2.4.4.1	Inputs to Code and Debug	3-55
3.2.4.4.2	Program Coding and Debug Development	3-56
3.2.4.4.3	Outputs from Code and Debug	3-57
3.2.4.5	Checkout Testing and Integration	3-57
3.2.4.5.1	Inputs to Checkout Testing and Integration	3-58
3.2.4.5.2	Checkout Testing and Integration Procedure	3-58
3.2.4.5.3	Outputs from Checkout Testing and Integration	3-60
3.2.4.6	Qualification Testing	3-61
3.2.4.6.1	Qualification Testing Inputs	3-61
3.2.4.6.2	Development of Qualification Tests	3-62
3.2.4.6.3	Output from Qualification Testing	3-63
3.2.4.7	Qualification Review Support	3-63
3.2.4.8	LBEF Support	3-63
3.2.4.9	Shipboard Qualification Support	3-64
3.2.4.10	Simulation and Test Program Development	3-65
3.2.4.11	Operators Manuals	3-68
3.2.4.12	Configuration Management	3-69
Section 4	Program Testing	4-1
4.0	INTRODUCTION	4-1
4.1	Conventional Program Testing	4-1
4.1.1	Program 1	4-3
4.1.1.1	Testing Objectives	4-3
4.1.1.2	Responsibility	4-4
4.1.1.3	Scope and Extent	4-5

TABLE OF CONTENTS (continued)

<u>Paragraph</u>	<u>Title</u>	<u>Page</u>
4.1.1.4	Levels of Testing	4-6
4.1.1.5	Test System Viability	4-6
4.1.1.6	Design for Ease of Testing	4-7
4.1.1.7	Methods and Tools	4-8
4.1.1.8	Test Documentation	4-10
4.1.1.8.1	Test Documentation for the Common Program	4-10
4.1.1.8.2	Test Documentation for the Navigation Program	4-11
4.1.2	Program 2	4-11
4.1.2.1	Testing Objectives	4-12
4.1.2.2	Responsibility	4-13
4.1.2.3	Scope and Extent	4-14
4.1.2.4	Levels of Testing	4-15
4.1.2.5	Test System Viability	4-15
4.1.2.6	Design for Ease of Testing	4-16
4.1.2.7	Methods and Tools	4-16
4.1.2.8	Testing Documentation	4-18
4.2	Top-down Concept Program Testing	4-20
4.2.1	Program 3	4-22
4.2.1.1	Testing Objectives	4-22
4.2.1.2	Responsibility	4-23
4.2.1.3	Scope and Extent	4-24
4.2.1.4	Levels of Testing	4-25
4.2.1.5	Test System Viability	4-25
4.2.1.6	Design for Ease of Testing	4-26
4.2.1.7	Methods and Tools	4-27
4.2.1.8	Testing Documentation	4-29
4.2.2	Program 4	4-29
4.2.2.1	Testing Objectives	4-33
4.2.2.2	Responsibility	4-33

TABLE OF CONTENTS (continued)

<u>Paragraph</u>	<u>Title</u>	<u>Page</u>
4.2.2.3	Scope and Extent	4-34
4.2.2.4	Levels of Testing	4-35
4.2.2.5	Test System Viability	4-36
4.2.2.6	Design for Ease of Testing	4-36
4.2.2.7	Methods and Tools	4-38
4.2.2.8	Testing Documentation	4-39
4.3	Conclusions and Recommendations	
Section 5	Software Quality Assurance (Task 2.4)	5-1
5.0	INTRODUCTION	5-1
5.1	Software Quality Assurance Program	5-1
5.1.1	Objectives	5-2
5.1.2	Procedures	5-2
5.2	Design Control	5-3
5.2.1	Objectives	5-5
5.2.2	Procedures	5-5
5.3	Development Controls	5-6
5.3.1	Objectives	5-6
5.3.2	Procedures	5-7
5.4	Testing/Certification Controls	5-10
5.4.1	Objectives	5-10
5.4.2	Procedures	5-11
5.5	Program Records and Change Controls	5-18
5.5.1	Objectives	5-18
5.5.2	Procedures	5-19
5.6	Technical Organization Responsibility	5-21
5.7	Summary and Conclusion	5-25
Section 6	Software Configuration Management	6-1
6.0	INTRODUCTION	6-1
6.1	Description of a Representative Software Configuration Process	6-2

TABLE OF CONTENTS (continued)

<u>Paragraph</u>	<u>Title</u>	<u>Page</u>
6.1.1	Configuration Control of Computer Programs	6-4
6.1.2	Computer Program Library (CPL)	6-5
6.1.3	CPPR/CPCO Processing	6-8
6.1.4	SCCB Processing	6-9
6.1.5	Baselines	6-10
6.1.6	Configuration Audits	6-12
6.2	Program 1 Software Configuration Management	6-12
6.2.1	Description of Program 1 SCM	6-12
6.2.1.1	Physical Configuration Audits	6-14
6.2.1.2	Program Verification Audits	6-14
6.2.1.3	Informal Audits	6-15
6.2.1.4	Formal Audits	6-16
6.2.1.5	Audit Reports	6-16
6.2.2	Effectiveness of Program 1 SCM	6-16
6.3	Program 3 Software Configuration Management	6-17
6.3.1	Description of Program 3 SCM	6-17
6.3.1.1	Formal SCM	6-17
6.3.1.2	Informal SCM	6-19
6.3.2	Effectiveness of Program 3 SCM	6-20
6.4	Program 4 Software Configuration Management	6-21
6.4.1	Description of Program 4 SCM	6-21
6.4.2	Effectiveness of Program 4 SCM	6-23
6.5	Common Program Software Configuration Management	6-23
6.5.1	Description of Common Program SCM	6-23
6.5.2	Effectiveness of Common Program SCM	6-24

TABLE OF CONTENTS (continued)

<u>Paragraph</u>	<u>Title</u>	<u>Page</u>
Section 7	Real-time Automatic Casualty Recovery (Task 2.6)	7-1
7.0	INTRODUCTION	7-1
7.1	Program 1	7-2
7.1.1	Mission Requirements	7-2
7.1.2	AN/UYK-7 Computer Attributes	7-3
7.1.3	Recovery Design Approach	7-4
7.1.4	How Success was Achieved	7-16
7.2	Program 2	7-18
7.2.1	Mission Requirements	7-18
7.2.2	AN/UYK-7 Computer Attributes	7-19
7.2.3	Recovery Design Approach	7-21
7.2.4	How Success was Achieved	7-25
7.3	Program 3	7-27
7.3.1	Mission Requirements	7-27
7.3.2	AN/UYK-7 Computer Attributes	7-29
7.3.3	Recovery Design Approach	7-32
7.3.4	How Success was Achieved	7-37
7.4	Conclusions	7-39
Section 8	Real-time On-line Maintenance Testing (Task 2.7)	8-1
8.0	INTRODUCTION	8-1
8.1	Confidence Testing Detailed Descriptions	8-6
8.1.1	Program 1 Confidence Tests	8-6
8.1.2	Program 2 Confidence Tests	8-6
8.1.3	Program 3 Confidence Tests	8-6
8.1.3.1	General Control	8-6
8.1.3.2	Unique Implementation Features	8-7
8.1.3.3	Tools and Methodology for Program 3 Confidence Tests Development	8-8
8.2	Diagnostic Testing in Reduced Configura- tion Mode	8-9

TABLE OF CONTENTS (continued)

<u>Paragraph</u>	<u>Title</u>	<u>Page</u>
8.2.1	Program 1 Diagnostics	8-9
8.2.2	Program 2 Diagnostics	8-9
8.2.3	Program 3 Diagnostics	8-10
8.3	System Interface Testing	8-11
8.3.1	Program 1 System Interface Tests	8-11
8.3.2	Program 2 System Interface Testing	8-11
8.3.3	Program 3 System Interface Tests	8-13
8.4	System Operability Testing	8-13
8.4.1	Program 1 Operability Testing	8-14
8.4.2	Program 2 Operability Testing	8-14
8.4.3	Program 3 Operability Testing	8-14
8.5	Overview on Tools and Methods of Test Program Development	8-15
8.5.1	Top-Down Test Software Development	8-15
Section 9	Host Development Support Software (Task 2.8)	9-1
9.0	INTRODUCTION	9-1
9.1	Usage of Development Support Software and Tools	9-2
9.2	Descriptions of Support Tools	9-6
9.2.1	AN/UYK-7 Simulator	9-6
9.2.2	Automated Common Program Certification Test Support Software	9-6
9.2.3	CMS-2 System Tape Generator (SYSMAKER)	9-8
9.2.4	CMS-2Y	9-8
9.2.5	Cost/Schedule Technical Achievement Reporting System (CSTAR)	9-10
9.2.6	Program Management Information System (PROMIS)	9-10
9.2.7	Design Object Drawing (DOD)	9-11
9.2.8	Dynaprobe	9-11
9.2.9	Exec-8 Tools	9-13

TABLE OF CONTENTS (continued)

<u>Paragraph</u>	<u>Title</u>	<u>Page</u>
9.2.10	Flow Charts	9-13
9.2.11	Fortran V	9-14
9.2.12	Graph	9-14
9.2.13	Library	9-14
9.2.14	Log CMS-2 Tasks	9-15
9.2.15	Mapper	9-15
9.2.16	Management Responsibility Reporting (MRR)	9-16
9.2.17	Processing Evaluation Tool (PET)	9-16
9.2.18	RIPS*.BLDSIT	9-17
9.2.19	RIPS*.COMPAR	9-17
9.2.20	RIPS*.CORE	9-21
9.2.21	RIPS*.ED	9-21
9.2.22	RIPS*.MANPOWER	9-21
9.2.23	Runstream Generator (RIPS*.RSG)	9-22
9.2.24	RIPS*.RUN	9-25
9.2.25	RIPS*.SYSIN	9-25
9.2.26	RIPS*.SYSOUT	9-25
9.2.27	RIPS*.TAPED	9-25
9.2.28	RIPS*.TAPIN	9-26
9.2.29	RIPS*.TAPOUT	9-26
9.2.30	Trace Input Process Output (RIPS*.TIPO)	9-26
9.2.31	Structured Narrative	9-28
9.2.32	ULTRA 32 Macro Assembly System	9-28
9.2.33	Utility Package (UPAD)	9-28
9.2.34	Debug Aids	9-36
9.2.35	Wrap-Around Simulation	9-37
9.3	Software Development Tools Effectiveness	9-38
9.3.1	Core Reports (RIPS*.CORE)	9-38
9.3.2	CMS-2 Monitor	9-38

* - Information on Remote Integrated Programming System (RIPS)
is Company Proprietary to Sperry Univac.

TABLE OF CONTENTS (continued)

<u>Paragraph</u>	<u>Title</u>	<u>Page</u>
9.3.3	Flowcharts	9-39
9.3.4	Function Flow Diagrams	9-39
9.3.5	Host Tools	9-40
9.3.6	Log CMS-2 Tasks	9-40
9.3.7	Program Compilation	9-40
9.3.8	Program Management Tools	9-41
9.3.9	RIPS.TIPO	9-41
9.3.10	Runstreams	9-42
9.3.11	Scenario Controlled Testing	9-42
9.3.12	Structured Narrative	9-43
9.3.13	System Generation	9-44

APPENDICES

<u>Appendix</u>	<u>Title</u>
A	AN/UYK-7 Computer Overview
B	Overview of Software Guidelines for Users of the Program 1 Common Program
C	Overview of Program 2 Maintenance Investigation Engineering Report
D	Overview of Program 2 Simulation Program Working Papers
E	Overview of the System Software Management Plan for Program 3 and 4.
F	Overview of the System Software Standards and Conventions for Programs 3 and 4.
G	Overview of the Executive Operation System User's Reference Manual for Programs 3 and 4.
H	Glossary of Terms
I	Bibliography

LIST OF TABLES

<u>Number</u>	<u>Title</u>	<u>Page</u>
1-1	Delivered Documentation	1-25
1-2	Size and Complexity	1-26
1-3	Program Manning	1-33
8-1	Program 1 Confidence Testing	8-2
8-2	Program 3 Confidence Tests	8-3
8-3	Program 3 System Interface Test	8-4
8-4	Program 1 Operability Tests	8-5
8-5	Program 2 Operational Testing	8-6
9-1	Development Support Software and Tools Usage	9-3
9-2	Formal Constructs	9-29
9-3	Design Object Summary	9-35

LIST OF FIGURES

<u>Number</u>	<u>Title</u>	<u>Page</u>
1-1	Major Contract Milestone Schedule	1-28
1-2	Manpower Schedules	1-29
2-1	Program 3 Development Schedule	2-7
2-2	Program 4 Development Schedule	2-8
2-3	Top-Down Hiearachies	2-13
2-4	Subrouting in a Top-Down Hierarchy	2-14
2-5	Design Object Symbol	2-15
2-6	Design Object Replication	2-16
2-7	Detached Hierarchy	2-17
2-8	Design Object Summary Table	2-18
2-9	Dead Reckning Module Design Object Drawing	2-19
2-10	Data Converter Module Design Object Drawing	2-20
2-11	Production Status Report Forms	2-24
3-1	Chan of Documentation	3-32
3-2	Major Tasks and Schedule for Program 4 Software and System Integration Task	3-34
3-3	Program 4 Work Package Form	3-35
3-4	Bi-Weekly Status Report Form	3-36
3-5	Status Report Form	3-37
3-6	Simulation and Testing Functional Interfaces	3-37
4-1	Typical Checkout Plan Sheet	4-30
4-2	Program Checkout Procedure	4-31
4-3	Checkout Log Sheet	4-32
4-4	Simulation and Testing Functional Interfaces	4-40
5-1	Typical Sperry Univac DSD Functional Organization	5-4
5-2	Typical Sperry Univac DSD Project Organization	5-4
5-3	Technical Areas of Responsibility	5-23

LIST OF FIGURES (continued)

<u>Number</u>	<u>Title</u>	<u>Page</u>
6-1	Representative Sequence for Processing Change Requests	6-11
7-1	AN/UYK-7 Computer Architecture for Program 1	7-4
7-2	Software Interpretation of AN/UYK-7 CPU Interrupts	7-9
7-3	AN/UYK-7 Computer Architecture for Program 2	7-20
7-4	AN/UYK-7 Computer Architecture for Program 3	7-31
7-5	Central Computer Complex Operational Configurations and Transitions Diagram	7-34
9-1	Simplified Schematic of Scenario Usage	9-7
9-2	Dynaprobe - 7900/DYNAPAR Functional Flow	9-12
9-3	Sample Timing Data	9-18
9-4	Sample Timing Data	9-19
9-5	Sample Timing Data	9-20
9-6	Manpower Backlog	9-23
9-7	Manpower Backlog Impact	9-24
9-8	Example of Structured Narrative	9-31
9-9	Program Listing Corresponding to the Example for Structured Narrative	9-32
9-10	Design Object Drawing	9-34
E-1	Relationship of the Software Management Plan with Other Documents	E-2

LIST OF ABBREVIATIONS AND ACRONYMS

ASCII
American Standard Code for Information Interchange

AUTO
Automatic

C and CS
Command and Control System Operational Program

CCB
Change Control Board

CDI
Casualty Diagnostic Interface (Module) of the Common Program

CDR
Critical Design Review

CEP
Common Error Processing

CI
Configuration Item

CM
Configuration Management

CMS-2
Compile Monitor System

COM
Computer Output Microfilm

CP
Common Program

CPCI
Computer Program Configuration Item

CPCO
Computer Program Change Order

CPDS
Computer Program Design Specification

CPL
Computer Program Library

CPOM
Computer Program Operator's Manual

CPP
Computer Program Package

CPPR
Computer Program Problem Report

CPPS
Computer Program Performance Specification

CPU
Central Processing Unit

CPUA
Central Processing Unit A, opposed to B of the same computer.

CPUB
Central Processing Unit B

CPTPL
Computer Program Test Plan

CPTPR
Computer Program Test Procedure

CSARITH
Common System Arithmetic Test (routine of the Common Program)

CSCPU
Common System CPU monitor test (routine of the Common Program)

CSDD
Computer Subprogram Design Document

CSIT
Computer System Interface Test

CSMIOT
Common System Memory, Input/Output Test (routine of the Common
Program)

CSOT
Computer System Operational (Operability) Test

D/A
Digital/Analog

DCN
Document Change Notice

DOD
Design Object Drawing

DODO
Design Object Drawing Outputter

DSD
Defense Systems Division

DRO (Memory)
Direct Read Out

ECP
Engineering Change Proposals

ESR
Executive Service Request

FAT
Factory Acceptance Test

FCN
Field Change Notices

GFE
Government Furnished Equipment

GFI
Government Furnished Information

HZ
Hertz, Cycles Per Second

IDS
Interface Design Specification

INSTR.
Instruction

I/O
Input/Output

IOA
Input-Output Adapter

IOC
Input-Output Controller

IOC/IOA
Input-Output Controller/Adapter

IPR
In-Process Review

ISM
Interim Software Maintenance

LBEF
Land Based Evaluation Facility

MC
Monitor Console

MCC
Monitor Control

MIN
Minute

MMU
Mass Memory Unit

MOD X2
Model X (experimental) 2 (revision)

MRR
Management Responsibility Reporting

MSEC
Millisecond, 10^{-3} second
N/A
Not Applicable
NDRO
Non Destruction Read Only Memory
NTDS
Navy Tactical Data System
PCA
Physical Configuration Audit
PDR
Preliminary Design Review
PDS
Program Design Specification
PET
Processing Evaluation Tool
PM
Performance Monitoring Module
PML
Program Maintenance Log
PPS
Program Performance Specification
POFA
Performance Operability Function Appraisal (Test)
PP
Program Package
PPS
Program Performance Specification
PROMIS
Program Management Information System

RR
Reconfiguration Routines

RIPS
Remote Integrated Programming System

SC
Software Configuration Module

SCCB
Software Change Control Board

SCM
Software Configuration Management

SDC
Signal Data Converter

SDI
Software Development Instruction

SEC
Second

SHAPM
Ship Acquisition Project Manager

SL
System Loader (module of Program)

SOW
Statement of Work

SQA
Software Quality Assurance

STR
Software Trouble Report

TE AND I
Test, Evaluation and Integration

TIPO
Trace Inputs Processing Outputs

UPAK

Utility Package (module of the Common Program)

USEC

Microsecond, 10^{-6} second

VSS

Video Signal Simulator

WBS

Work Breakdown Structure

WS - 8506

Weapons Specification - 8506 (NAVORD) documentation standard
for computer programs.

EVALUATION

This report describes the software development technology and management practices employed on four specific developments by Sperry Univac.

The intent of the RADC program to which this document relates is to describe and assess software production and management tools and methods which significantly impact the timely delivery of reliable software.

The study contract is one of a series of six with different firms having the similar purpose of describing a broad range of techniques which have been found beneficial.

RADC is engaged in promoting utilization of Modern Programming Technology, also called Software Engineering, especially in large complex Command and Control software development efforts.

Roger W. Weber

ROGER W. WEBER
Project Engineer

STUDY REPORT SUMMARY

This Modern Programming Practices (MPP) study provides RADC descriptions of practices proven effective on software development. These practices were analyzed and compared for three large-scaled software development projects. On these projects Sperry Univac developed four programs identified by 1, 2, 3 & 4. The resulting report information is intended to benefit both RADC and Sperry Univac by documenting proven software technology.

The described practices were shown effective for developing software:

- a) Within cost and schedule.
- b) With greater technical and managerial control.
- c) With greater customer and managerial visibility.
- d) For simplified use and maintenance by other agencies.

Analysis of the studied program developments basically show:

- a) Top-down Program Development
 - . Implemented with modern tools cut cost by 15%.
 - . Can produce reliable software in less time.
- b) Manning software development, generally required:
 - . 10% for analysis
 - . 30% for design
 - . 35% for coding and debugging.
 - . 25% for testing.

SUMMARY (continued)

- c) Testing real-time programs was effective using:
 - . Scenario test control.
 - . Wrap-around simulation.
 - . On-line real-time data extraction/recording.
 - . Computer-aided data reduction.
- d) Controlled development produced reliable software.
 - . Applying real-time testing.
 - . Continuing quality assurance.
 - . Applying formal configuration management.
 - . Has demonstrated error-free shipboard use of Program 1 for over six months.
- e) Software can be fault tolerant by providing:
 - . Automatic recovery of first level hardware failures.
 - . On-line component maintenance.

The major intent of the study was to describe Sperry Univac's software development technology and management practices germane to MPP. The separate sections of this technical report are summarized as follows:

a) SECTION 1.

Program Environment - The development environment for each program is described. The description includes: Magnitude of each program, the hardware environment, special software, delivered items, schedule, manpower, and personnel profile.

b) SECTION 2.

Top-down Program Development - This method of implementing software is described in detail. The description includes the top-down development procedures, and constructs. The information is presented in detail to allow a developer to use the top-down program development method for implementing software.

c) SECTION 3.

Software Development Standards - The software development standards used by Sperry Univac are described for each program. The description includes each phase of implementing the programs.

d) SECTION 4.

Program Testing - A description of the testing method used for each program is included. The description includes the testing objectives, responsibility, scope, levels, test viability, testing ease, methods, and testing documentation.

e) SECTION 5.

Software Quality Assurance - The Software Quality Assurance (SQA) that was employed by Sperry Univac is described for program 3. The description includes the objectives, procedures, design control, develop-

e) SECTION 5. (continued)

ment control, testing/certification control, and configuration control.

f) SECTION 6.

Software Configuration Management - The Software Configuration Management (SCM) methods that were used and their effectiveness are described for Programs 1, 3, and 4.

g) SECTION 7.

Real-time Automatic Casualty Recovery - Selected elements of Programs 1, 2, and 3 are described. The descriptions that are provided pertain to the elements that safeguard the loss of the overall mission critical functions. Particular emphasis is given toward the description of the recovery from the loss of a computer hardware module.

h) SECTION 8.

Realtime On-line Maintenance Testing - A description is provided for the real-time on-line maintenance testing that was performed for programs 1, 2, and 3. The description includes:

- 1) On-line confidence tests for execution in the background relative to the other real-time application programs.
- 2) Provisions for device sharing to continue real-time operation while executing the equipment diagnostic test programs.

h) SECTION 8 (continued)

- 3) The test capabilities of the operating systems and other test software to provide automated system interface tests and run-time operability tests.

i) SECTION 9

Host Development Support Software - Descriptions are provided for the developmental support software tools that were utilized in developing the programs. A measure of cost effectiveness through manpower savings is shown for Programs 2, 3, and 4.

j) APPENDIX SECTION

Several appendices are included. These appendices provide additional information in support of the other report sections. The Appendix section also includes the glossary of terms and definitions.

SECTION 1
PROGRAM ENVIRONMENTS (Task 1.2)

1.0 INTRODUCTION

This section discusses the environment of software development at Sperry Univac in terms of four successful large real-time command and control programs completed for the U.S. Navy. This section reviews each program's magnitude, operating system(s), hardware suite(s), special requirement and deliverable items. It summarizes schedule, manning, and personnel attributes.

Within limitations of available data, this section includes information on program compilation/assembly, turnaround time, and equipment availability/accessibility to programmers.

The four programs reflect the evolution of modern programming practices at Sperry Univac over a seven year period. Each program involved modification and expansion of existing available programs, and generation of new programs. Each program was separately documented in conformance with standards set by the Navy. Each used a common set of software for its baseline operating system (common program), and each operated on the Navy's standard large central computer, the AN/UYK-7. Appendix A presents a brief technical description of this computer. Each operating system required adaptation to the applications unique computer/peripheral configurations.

A summary of the magnitude of each program follows:

Program	No. Lines of Source Code Generated	No. Pages of Documentation in Final Edition	No. of Man Months
1	90,000	8,059	630
2	500,000	27,014	2,960
3	26,600	3,507	430
4	13,150	2,259	115

1.1 PROGRAM DESCRIPTIONS

The Program 1 navigational program, Program 2 application programs, and Program 4 were entirely developed for the specified applications. The other programs were built upon the most advanced existing common programs available as a baseline program.

1.1.1 Program 1

Software was specified, designed, developed, and certified by formal testing to provide a real-time Command and Control System (C&CS). C&CS included the operating system software for integrated use by two real-time application programs; a navigational program produced as a second part of this program and an application program produced by another contractor.

1.1.1.1 Operating System

The operating system software consisted of a single program with capabilities to:

1.1.1.1 (continued)

- a) Allocate computer processing time to the various task programs.
- b) Handle supervisory control and interface for standard data processing peripherals.
- c) Generate and debug programs for delivery using the target computer.
- d) Compute commonly used mathematical and conversion functions.

The operating system software optimized use of computer hardware features by:

- a) Processing executive state interrupts.
- b) Initiating I/O in the processor executive state.
- c) Executing privileged instructions in the processor executive state.

Design goals for the operating system software included:

- a) Quick response to task program requests.
- b) Automatic recoverability from hardware component failures and software errors.
- c) Allowing for system growth with minimum impact on existing programs.
- d) Minimizing interference between task programs in a multi-programmed and multiple processor environment.

The program design for the operating system and application software required assignment of functions to related tasks and task combinations which created the following seven modules:

- 1) Common Control (CC) - the executive.
- 2) Common Systems (CS) - system data, shared routines, automatic recovery.

1.1.1.1 (continued)

- 3) Common Peripheral (CP) - standard interface to data processing equipments.
- 4) Dynamic Module Replacement (DMR) - allow dynamic module segment insertion and deletion.
- 5) Processor Confidence Tests (CT) - periodic processor check.
- 6) Debug (DB) - on-line programming debugging tool.
- 7) Utility Package (UPAK) - off-line program maintenance aid.

Modules communicated through intermodule messages and/or through shared data. Intermodule message use was restricted to transferring data which required immediate processing by the receiving module, or to transferring data which was required by only one or two other modules.

The Operating System administered the scheduling of modules and intermodule communication. In each module, a preamble table and a segment linkage table provided a standard module/executive interface. The preamble table provided linkage and attributes for the module tasks and for location of module task history stores. The segment linkage table contained the identification and base addresses of program segments which were directly addressable by the tasks of the module.

Communication between the modules and the executive was performed by means of an Executive Service Request (ESR); the ESR provided a method for modules to communicate service requests and operational information to the Operating System. By communicating requests for the execution of privileged operations and by transmitting operational activity information, the ESR enabled the system to provide both information and control in the executive state. An Interrupt Status Code (ISC) was specified as part of the ESR instruction format and identified the specific executive function requested. A list of ESR names,

1.1.1.1 (continued)

function descriptions and associated parameters were included in the Software Guidelines (see appendix B). ESRs were provided for I/O control, module status modification, preregistration of I/O related interrupt processing, and intermodule message transmission.

In the C&CS, three levels of data (local, regional, and system) were provided to facilitate the software development process and to protect fundamental operating data.

1.1.1.2 Navigation Program

The Navigation Program was a task state application program that operated under the real-time operating system (i.e., Common Program of Program 1). It provided the computational functions required by the navigation equipment for collecting, interpreting, and supplying navigation information (e.g., ownship position, attitude, speed, and heading) for subsequent system use. The program consisted of the following modules:

- a) Resident Control - provided local supervisory functions.
- b) Inertial Navigation - accepted navigational data from the equipment and generated velocity and positional data for output to the system operator and to a digital-to-analog converter. The program provided feedback to the inertial navigation equipment to maintain platform stability.
- c) Dead reckoning - generated a best reasonable estimate of velocity and positional data (independent of the navigation equipment) on the basis of the ships heading, speed, and last known position.

1.1.1.2 (continued)

- d) Satellite Navigation - determined ownship position from information obtained from a maximum of eight navigation satellites. The position fixes were supplied to the operator and used to reset the navigation equipment and to correct errors in ship position. The program could also reprocess a calculation with operator edit of the data.
- e) Keyboard/Printer Interface - provided the service and scheduling required by the navigation program for man-machine interface.

1.1.1.3 Navigation Simulation Program

The navigation simulation program was a task state program that operated under the real-time operating system software. This program simulated navigation equipment generated signals and responded to navigation program outputs.

The program provided closed loop control when it simulated an automatic interactive device with the navigation program and open loop control when no feedback was provided by the simulated navigation program environment. The program accepted control from key-ins at either of two keyboard printers and from paper tape emulation of the keyboard entries.

The program provided for printouts from either the keyboard printer or the high speed printer to record:

- a) Status of the simulation program showing mode, readiness and abnormal conditions.
- b) Simulated position, velocity and attitude as it changed.
- c) Differences between predicted and actual extracted navigation program generated values.
- d) Status and error summary reports, upon operator demand.

1.1.1.3 (continued)

When co-resident with the operational program, the simulation program interfaced the navigation program by simulating computer I/O internally via shared data or externally via end-around inter computer I/O channels.

The simulation program was used for acceptance testing of the navigation program before delivery and by the prime contractor to perform integration testing with the line equipment. After enhancement, the program provided real-time simulation for endurance and operational readiness tests in the Land Based Test Facilities.

1.1.1.4 Reduced Capability Program

The reduced capability program operated with a minimal subset of the full hardware suite and provided a minimal subset of system functions, including all mission-critical functions.

The reduced capability program operated within the AN/UYK-7 components of one processor, three 16,348 word memory units, one or two I/O controllers, and a proper subset of the interfaceable peripheral devices. Hardware components critical to the mission functions, were provided as isolateable redundant or alternate components. The reduced capability program accommodated automatic and operator directed modification of the hardware component suite.

The reduced capability program allowed for software reconfiguration within the AN/UYK-7 such that:

- a) Computer hardware diagnostic programs could be executed using hardware components isolated by software from the real-time program.
- b) Power to either of the two computer mainframes could be off in order to allow hardware replacement of a failed component.

1.1.2 Program 2

Software was specified, designed, developed and certified by formal testing under simulation to provide a large centralized integrated Command and Control System (C&CS). All software of the operating system, operational application programs, test programs, and simulation programs for use in the central computer suite was developed under Program 2.

1.1.2.1 Operating System

The operating system was based upon the advanced operating system from Program 1 at the start of the Program 2 development. This baseline program was expanded in capacity by expanding control tables, and enhanced in capability by adding new I/O interfaces and new functions. Since the central computer for Program 2 was not fully hardware integrated, two operating system designs were developed using a single program source; one allowing dual CPU multiprocessing and the other allowing single CPU processing. Each operating system supplied its own I/O interfaces as determined by the hardware suite design. This dual operating system design shared use of certain computer memory units, I/O handlers and an inter-computer I/O channel which provided a degree of software integration to the two operating systems.

1.1.2.2 Test Program

Enhanced Program 1 test software was used to formally certify all Program 2 Operating System functional and performance capabilities.

1.1.2.3 Operational Program

Concurrent with Operating System development, a separate working group developed the modular task software which when integrated with the operating system software in the central computer, provided the C&CS operational program. This provided command per-

1.1.2.3 (continued)

sonnel with the capability to continuously monitor the ships tactical mission environment and to control the weapon systems. The operational program was subjected to simulated operational testing before acceptance as conforming to the C&CS functional and performance specifications.

1.1.2.4 Simulation Program

The simulation program was modular and ran under the Program 2 Operating System, but in a separate AN/UYK-7 computer. It simulated the real-world environment and the weapons system via intercomputer I/O interfaces and operator generated scenario.

The resulting system provided simulation and recordings for acceptance of the operational program, and provided a basic C&CS training tool.

The Program 2 simulation program provided the following capabilities:

- a) Missile Fire Control Simulation, as it affected the operational program. This included simulation of combined data from a Guided Missile Launching System, Missiles, and a Missile Radar.
- b) Gun Fire Control Simulation, as it affected the operational program. This included simulation of the combined data from two gun mounts, a Surface radar, and an Air Tracking Radar. The control operator was able to include both Missile Fire Control and Gun Fire Control simulations in the same exercise.
- c) Underwater Fire Control Simulation, as it affects the operational program. This included simulations of the combined data from sonar, torpedo tubes, and a Guided Missile Launching system. In this case, an additional intercomputer channel was required.

1.1.2.4 (continued)

- d) Track Generation Simulation provided generation, maintenance, alteration, or deletion of tracks on a display, and simulated the presence of live radar data to the operational program.
- e) Sensor Interface Data System Simulation, based on the simulated track environment as it affects the C&CS. This included simulation of the combined data from Air Search Radars, a Surface Search Radar, a Beacon Video Processor, and a Digital Video Processor. Sensor simulation consisted of simulating the interface between the Sensor Interface Data System computer and the C&CS. This was implemented via shared memory and inter processor interrupt communications from the Simulation computer and C&CS Computer.
- f) Signal Data Converter Simulations, as it affected the operational program. This included:
 - 1) Accepting control signal data from C&CS.
 - 2) Accepting search radar designation and course order data, each requiring digital-to-synchro conversion, from C&CS.
 - 3) Accepting data intended for digital output channels from C&CS.
 - 4) Accepting interrupts and data intended for control of the Signal Data Converter from C&CS.
 - 5) Providing status, digital output, synchro, digital input channel, and sensors, and ownship motion equipment.
 - 6) Functionally simulating ownship motion, radar, wind equipment, and target designation transmitters.

1.1.2.4 (continued)

- g) A Data Terminal Set as it affected C&CS. This included simulation of a multi-participating communications net based on the simulated tracks.
- h) Simulation of a Data Communications Set on a communications system, as it affected C&CS. This included simulation of multiple communications equipped interceptors.
- i) Satellite Navigation System Simulation, as it affected the Operational Program. This included auxiliary console inputs and simulated satellite orbits.

The simulation program utilized a special Monitor Control Console to provide:

- 1) Keyboard data inputs to the Simulation Program.
- 2) Tabular display of status and data.
- 3) A primary means of operator control using an easy-to-operate peripheral.

Data was displayed on a CRT and entry was made via an alphanumeric keyboard. Simulation program operation was allowed from two different types of display consoles.

The simulation program display/operator interfaces included:

- a) Full control of the Simulation Program.
- b) A comprehensive set of displays showing status, and also the quantitized and positional data being transmitted to, or received from the C&CS Program by the Simulation Program.
- c) Sufficient operator inputs to allow sensitive control of the simulation process.
- d) A display of all simulated tracks.

1.1.3 Program 3

Program 3 comprised a set of specialized system functions which expanded the Navy furnished baseline MOD X2 Common Program to provide the real-time operating system for a large integrated C&CS.

These specialized functions provided:

- a) Centralized system loading, system initialization, system casualty recovery, and reconfiguration functions.
- b) A centralized common interface for input/output of inter-computer channel data shared by multiple application subsystems.
- c) Centralized management of the system shared data.
- d) Centralized, semiautomatic control of on-line C&CS hardware confidence tests and manual control of computer diagnostics and peripheral performance tests without detracting the available computer and peripheral resources from the real-time application programs.
- e) Centralized control for standard peripheral devices including interface programs for a keyboard/display, and a variety of different keyboard printers, magnetic tape subsystems, and disk memory subsystems.

A formal acceptance test for Program 3 verified proper operation of the specified functions.

1.1.4 Program 4

Program 4 was a real-time application subprogram which used the Program 3 operating system software. It was the software component for a complex process control system which featured both automatic and computer aided manual control. This program provided display outputs, logic functions and performance monitoring, performed calculations, and gave location assistance required for maintaining real-time operations.

1.1.4 (continued)

Development of Program 4 involved generation of the design specification, software acceptance test program, real-time program, program operator's manual, acceptance test plan and procedures, and subprogram design documents. The performance specification, and real-time simulation software for Program 4 were furnished by the customer.

1.2 PROGRAM BASELINING THROUGH SYSTEM TEST/EVALUATION

The certification Program development group of the C&CS department and later the separate Systems Evaluation department developed the Common Program system test and evaluation program. This development, like the operating system development, began each time with the most advanced evaluation program available, in order to generate the required program test of the revised Common Program. Since the resulting test program was operated in the development equipment suite under simulated real-time conditions, it too required an operating system.

The initial baseline Operating System Software used to develop the Real-time Operating System and Test/Evaluation System for each project is listed as:

<u>Program</u>	<u>Baseline Real-time OS</u>	<u>Baseline T/E System</u>
1	IR&D MODX0 CP, 1969	MODX0 T/E, 1969
2	Program 1, April 1973	Program 1, CP Cert. 1973
3	MODX2 CP, Nov, 1974	MODX2 CP Cert., Nov. 1974
4*	MODX2 CP, Sept. 1973	Not Applicable

*Program 4 was an application program and not an Operating System

1.2 (continued)

Where:

- CP - Common Program
- IR&D - Independent Research and Development
- T/E - Test and Evaluation
- Cert. - Certification Test Program

1.3 TARGET HARDWARE SUITES

Throughout a program's development, as more of the specified target equipments were furnished or as different target equipments were specified, the furnished hardware suite for program generation, test and evaluation changed.

Since three of the four studied programs included real-time operating system software, development of these programs were directly affected when the computer configuration and peripheral equipments were changed. Each time a hardware suite was modified, as a minimum, the program operation had to be checked out, the operator's manuals modified and the I/O handlers either modified or new I/O handlers added.

1.3.1 Program 1

The final target hardware suite for Program 1 was basically unchanged during system design. The three computer centers used during various stages of the program development contained identical or nearly-identical interfacing equipments.

The following list describes in generalized terms the hardware used during the software development.

- a) The central computer was the target AN/UYK-7 configured with two mainframes, two CPUs, six MMUs each containing 16,384 32-bit words of core memory, and two IOC/IOAs, each containing 16 NTDS I/O channels, all fully integrated by the bussing network.

1.3.1 (continued)

- b) One system control console provided was either the target device or a functionally compatible device which required only modification of the values and sequence of control codes.
- c) One navigation keyboard/printer provided was either the target device or a logically-equivalent, but slower keyboard/printer.
- d) One magnetic tape storage subsystem provided was either the target dual transport device or a functionally compatible 4-transport device which required only modification of the values and sequence of control codes.
- e) One magnetic disk storage subsystem provided was either the target 1-drive unit or one drive of a 4-drive unit.
- f) One synchro-digital data convertor provided was either the target device or an inter-computer channel interface to a second central computer.
- g) One dual (redundant) navigation subsystem provided was either target equipment or was an intercomputer channel interface to a second computer.
- h) One paper tape reader/punch device provided was either the target device or a functionally-compatible device which required only modification of the value and sequence of control codes.
- i) One high speed printer and card reader, provided for program compiling, assembly, and on-line/off-line test data recording.
- j) Many intercomputer I/O interfaces were provided to simulate absent devices for automatic system test and evaluation.

1.3.2 Program 2

As each target system equipment became available, it was installed and integrated into the furnished program development center. Two major centers, local and remote, were provided for program development and system integration. The local center, used for program development, contained only enough equipment to fully test the C&CS central computer program. The remote center, used for final program integration and acceptance testing, contained a fully-integrated C&CS system.

Hardware interfaced during the software development included:

- a) The central computer configured as the target AN/UYK-7 computer and contained varied internal configurations of four mainframes, three CPUs, 13 MMUs each containing 16,384 32-bit computer words of main memory and four IOC/IOAs each containing 16 NTDS I/O channels. The network integrated these components but did not allow for total memory sharing.
- b) Two system control consoles, each provided multiplex I/O to: keyboard/printer, a magnetic tape handler, a paper tape reader, and a paper tape punch. These consoles were configured as target equipment.
- c) Two additional magnetic tape storage subsystems each provided four magnetic tape transports.
- d) One magnetic disk storage subsystem provided by the target one-drive unit.
- e) Two Monitor Control Consoles providing multi-functional alphanumeric display and keyboard consoles with internal refresh memory configured as the target displays.

1.3.2 (continued)

- f) Twenty one multifunctional command consoles each providing planned position indicator display, direct read out displays and several forms of operator entry buttons, switches and rotary controls, configured as the target equipment. These consoles were interfaced by multiplexers and required periodic interrogation by the software.
- g) Two functionally equivalent high speed printers for use during system test and evaluation were not target equipment.
- h) Four intercomputer channel interfaces configured as per the target system.
- i) Two data communication terminals.
- j) One Analog/Digital - Digital/Analog Converter.
- k) One navigation device.

1.3.3 Program 3

Program 3 was developed in CMS-2 code using the RIPS/UNIVAC 1108 hosted system. The generated program listings and machine loadable tape of absolute program elements were taken to a Navy-furnished center and integrated with the baseline operating system. The resulting programs were debugged and tested under simulated real-time conditions listing the furnished center equipment in four milestone levels. Programs of each level were installed at six other remote sites for subprogram development use. Overall the software interfaces required for these equipment sites were:

1.3.3 (continued)

- a) Four fully integrated configurations of the AN/UYK-7 central computer containing:
 - 1) Two CPUs, two IOC/IOAs, and six MMUs of 16,384 words each,
 - 2) One CPU, one IOC/IOA, and three MMUs of 16,384 words each,
 - 3) Two CPUs, two IOC/IOAs, and 9 MMUs of 16,384 words each,
 - 4) Two CPUs, two IOC/IOAs, and 11 MMUs of 16,384 words each,
- b) Four versions of the system control console configured as:
 - 1) One multiplexed keyboard/printer, one two-transport magnetic tape subsystem, one paper tape reader, and one paper tape punch.
 - 2) One multiplexed keyboard/printer, one paper tape reader, and one paper tape punch,
 - 3) One alphanumeric display/keyboard,
 - 4) One keyboard/printer.
- c) Four versions of the magnetic tape storage subsystem configured as:
 - 1) One four-transport Navy standard subsystem.
 - 2) One two-transport system in the system control console.
 - 3) One UNIVAC 8C magnetic tape subsystem.
 - 4) One UNIVAC 12C magnetic subsystem.
- d) Two versions of the single-drive magnetic disk storage subsystem. Both required a similar interface although one provided doubled storage capacity.

1.3.3 (continued)

- e) Two versions of a high speed printer and card reader for use during system test and evaluation.
- f) Three inter-computer interfaces, a dual channel interface between the two C&CS Central Computers and one to connect another real-time computing center.

1.3.4 Program 4

Program 4 was developed using the same facilities as that for Program 3, but was installed into only two remote sites, one for the customer's acceptance test and the other for full C&CS integration. Program 4 did not directly interface any peripherals and was always run using a computer environment logically equivalent to the target computer, therefore, no significant effects were imposed on its development by various hardware suites.

1.4 SPECIAL REQUIREMENTS

Each studied software development was designed to accomplish many unique real-time requirements to be supported by software design, coding, and testing.

1.4.1 Program 1

Program 1 met the following special requirements:

- a) The system automatically detected, isolated and recovered from all first level hardware failures in the central computer and from peripheral failure.
- b) The system maintained Navigation accurately across main memory program reloads and program reconfigurations.
- c) The system supported a minimum time period to recovery from a hardware component failure of a few seconds.

1.4.2 Program 2

Program 2 met the following special requirements:

- a) The integrated C&CS central program maintained continuous support to the specified range of activity levels and operating modes without fault for an extended period.
- b) The C&CS program structure and operating system software allowed operator entries to efficiently reconfigure the on-line software for varying conditions of hardware, operational readiness.
- c) The testing during system integration allowed for real world simulation interface testing of the four interconnected computer systems so each may be operationally tested independently within the target equipment suite.

1.4.3 Program 3

Program 3 met the following special requirements:

- 1) Compliance with the customer's Software Development Standards and Conventions.
- 2) Operation in multiple states, reconfiguring automatically following hardware failure, or manually under operator direction, to ensure continuous real-time support of the critical C&CS functions.
- 3) The resulting operating system supported development of the application programs within six months after receipt of order (ARO) and was fully developed and tested twelve months ARO.
- 4) The development group supported simultaneously seven program test, development, and integration centers, each with different hardware and software configurations.

1.4.4 Program 4

Program 4 met the following special requirements:

- 1) Integration with the results of the Program 3 development before testing and delivery.
- 2) Full-cycle processing at 8Hz using less than 15% of a CPU.
- 3) Continuous real-time support of a function throughout software/hardware reconfigurations.

1.5 CUSTOMER FURNISHED DATA

Each program's development began with a furnished contracted statement of work and varying amounts of functional requirement documentation. The programs were furnished the equipment and other items called out in paragraphs 1.2.1 through 1.2.4 for use in program generation, debugging, and acceptance testing at a customer supplied facility.

1.5.1 Program 1

Equipment - An integrated suite including target computers, target peripherals addressed by the operating system, an intercomputer channel to a second target computer for simulation of the Navigation system devices, and data processing equipment (magnetic tape, I/O console, card reader and high speed printer).

Software - ULTRA/32 Assembler/Librarian. This support software was operational in the furnished equipment suite and was maintained by the customer.

Documentation - Contractual Statement of Work, High Level System Functional Requirements, and Documentation Standards.

1.5.2 Program 2

Equipment - An integrated suite including a target computer, target peripherals addressed by the C&CS operational program, an inter-computer channel between the two operating systems computers and data processing equipment (magnetic tape, I/O console, card reader, card punch and high speed printer). The program generation suite was separate from the test facility.

Software - ULTRA/32 Assembler/Librarian and CMS-2Y Compiler/Monitor/Librarian with SYSMAKER. This software was furnished for the program generation suite and maintained by the customer.

Documentation - Contractual Statement of Work, Tactical Operational Requirements, System Design Data Document, Interface Design Specification, and Documentation Standards.

1.5.3 Program 3

Equipment - An integrated suite including the target computer from Program 1, target peripherals from Program 1 which emulated the Program 3 operating system equipment, an inter-computer channel to a second computer for simulation of the target inter-computer interfaces, and data processing equipment (magnetic tape, high speed printer and card reader and card reader).

Software - No additional support software was furnished on this program. The CMS-2Y system supplied for Program 2 was made available to this program.

Documentation - Contractual Statement of Work, Software Development Standards and Conventions, High Level Functional Specification, System Design Data Document, Degraded Mode Study Report, and Documentation Standard.

1.5.4 Program 4

Equipment - Shared the suite of Program 3.

Software - Same as for Operating System Software developed as Program 3.

Documentation - Contractual Statement of Work, Computer Program Performance Specification, Interface Design Specification, and documentation the customer supplied for Program 3.

1.6 FURNISHED SUPPORT SOFTWARE

The Navy supplied support software consisting of a program language assembler and language compiler/monitor/librarian. These batch processing systems were ready for use in the equipment suite using the target computer. Although the assembler was available first, it was later integrated within the compiler monitor system.

Beginning with the development of Program 3 and 4, Sperry Univac supplied the Remote Integrated Programming System (RIPS)*. RIPS extended the capabilities of the Western Utility Computer Center (1100 series computers, peripherals and Executive 8 time sharing) facilities by using the Navy controlled CMS-2Y software resident in an AN/UYK-7 computer connected as a peripheral to the 1108 computer. All four programs eventually were maintained under formal configuration management in the RIPS system.

The following table identifies the support software and period of its use on each program.

* - Information concerning RIPS is Company Proprietary to Sperry Univac.

1.6 (continued)

FURNISHED SUPPORT SOFTWARE

PROGRAM	Assembler (Period)	Compiler (Period)	RIPS (Period)
1	ULTRA/32 (1/70-1/74)	CMS-2Y (8/71-6/73)	RIPS (6/75-4/76)
2	ULTRA/32 (2/72-4/73)	CMS-2Y (3/73-7/76)	RIPS (10/74-4/76)
3	Not Applicable	CMS-2Y (1/74-11/74)*	RIPS (11/73-4/76)
4	Not Applicable	CMS-2Y (4/74-12/75)*	RIPS (4/74-12/75)
* - Used only for object program loading.			

1.7 LOCAL AND REMOTE DATA PROCESSING

Three data processing methods, batch, remote batch, and time sharing were available for program generation, testing of algorithms and program maintenance. The following table presents the degree of use on each program.

Program	Batch		Remote Batch		Time Sharing	
	CMS-2	ULTRA/32	CMS-2	ULTRA/32	Exec.8	RIPS
1 Op.Sys.	9%	75%	0	0	0	16%
1 Nav.	100%	0	0	0	0	0
2	17%	28%	0	0	0	55%
3	2%	0	0	0	0	98%
4	2%	0	0	0	0	98%

The above does not include data processing for program debugging, checkout and formal testing.

Computer turn around time data was not available.

1.8 DELIVERED ITEMS

Each program resulted in delivery of an operational computer program package and its associated documentation. Table 1-1 presents the list of program documents delivered and shows the final edition's page count. Table 1-2 shows the amount of program coding produced for each of seven levels of complexity.

Table 1-1. Delivered Documentation

Program	Generalized Title	Navy Standard	Final Edition Page Count
1	Program Performance Spec.	NAVSHIPS-0011	467
	Program Design Plan	NAVSHIPS-0011	1798
	Program Package	NAVSHIPS-0011	N/A*
	Program Description	NAVSHIPS-0011	323
	Operating Procedure	NAVSHIPS-0011	165
	Certification Plan	NAVSHIPS-0011	184
	Certification Procedures	NAVSHIPS-0011	2093
	Simulation Program Package	NAVSHIPS-0011	N/A*
	Simulation Program Description	NAVSHIPS-0011	283
	Navigation Program Documentation	NAVSHIPS-0011	2746
			8059 pgs.
2	Program Performance Spec.	NAVMAT	1725
	Program Design Spec.	NAVMAT	15360
	Program Package		N/A
	Certification Plan/Procedures	NAVMAT	4920
	System Manual/Operating Manual	NAVMAT	4909
			27014 pgs.
3	Program Performance Spec.	WS-8506, Rev. 1	425
	Program Design Spec.	WS-8506, Rev. 1	1450
	Program Package	WS-8506, Rev. 1	N/A*
	Program Test Plan	WS-8506, Rev. 1	69
	Program Test Procedure	WS-8506, Rev. 1	1273
	Operating Procedure	WS-8506, Rev. 1	259
	Program Mgmt. & Perf. Plan	Not Applicable	43
	Program Conf. Mgmt. Plan & Proc.	Not Applicable	56
	Quality Assurance Plan	Not Applicable	31
			3666 pgs.
4	Program Design Spec.	WS-8506, Rev. 1	574
	Program Package	WS-8506, Rev. 1	N/A*
	Program Test Plan	WS-8506, Rev. 1	45
	Program Test Proc.	WS-8506, Rev. 1	553
	Operating Procedure	WS-8506, Rev. 1	63
	Subprogram Design Docs	WS-8506, Rev. 1	1024
			2259 pgs.

* - N/A = Not Applicable

BEST AVAILABLE COPY

Table 1-2. Size and Complexity

CATEGORY OF COMPLEXITY	PROGRAM 1		PROGRAM 2		PROGRAM 3		PROGRAM 4	
	CMS-2	ULTRA/32	CMS-2	ULTRA/32	CMS-2	ULTRA/32	CMS-2	ULTRA/32
SIMPLE TASK STATE ROUTINES (PRIORITY, MESSAGE, PERIODIC PROCESS)	12.0K	25.0K	321.2K				10.5K	
SIMPLE DATA DESIGNS (VARIABLES, FIELDS, TABLES, SYS-DD EQUALS, PARAMETERS, ETC.)	7.0K	20.0K	156.6K		1.4K	.1K	2.6K	
MORE COMPLEX DATA DESIGNS (FIELD OVERLAYS, JOB CONTROL STREAMS TO COMPILE LOAD AND GENERATE LIBRARIES)			17.4K		3.5K	1.6K	0.05K	
MORE COMPLEX TASK STATE ROUTINES (DEMAND ENTRANCE, I/O INTERRUPT ENTRANCE, CENTRALIZED ROUTINES AS IN REGIONAL CONTROLLERS AND MATH ROUTINES)	10.0K	5.0K		3.0K	2.9K	6.8K		
COMPLEX DATA DESIGNS (OVERLAY TABLES, COMPLEX INTERDEPENDENT TABLES, SPECIAL PURPOSE CONSTANTS, REENTRANT DATA, SHARED DATA)	5.0K	1.0K	0.5K	0.5K	0.4K			
SINGLE STATE EXECUTIVE ROU- TINES (CLASS 4 ONLY, CLASS 3 ONLY, CLASS 2 ONLY ROUTINES, DELAYED CASUALTY RECOVERY)				0.5K				
MULTIPLE STATE EXECUTIVE ROU- TINES (CLASS 1 RECOVERY, IMMEDIATE ENTRANCE RECOVERY ROUTINES, I/O RECOVERY ROU- TINES, ONLINE TRANSITION CONTROL ROUTINES, ETC.)		5.0K		0.3K	6.0K	3.9K		
TOTAL	34K	90K	56K	495.7K	4.3K	500K	14.2K	12.4K
							13.15K	13.15K
								0

1.9 MAJOR CONTRACT MILESTONE SCHEDULE

The major contract milestone schedule presented as figure 1-1 presents the development schedule followed by each of the four programs. This schedule was a part of the contractual statement of work and was made a part of the internal Sperry Univac project plan for each program. The major milestones represent significant completions of contractual requirements. By superimposing these schedules on the manpower schedules in figure 1-2, the reader will have a graphic representation of manpower required to complete each major milestone.

1.10 MANPOWER SCHEDULES

The manpower schedule is shown in figure 1-2.

The total man months of paid expended labor for each program is:

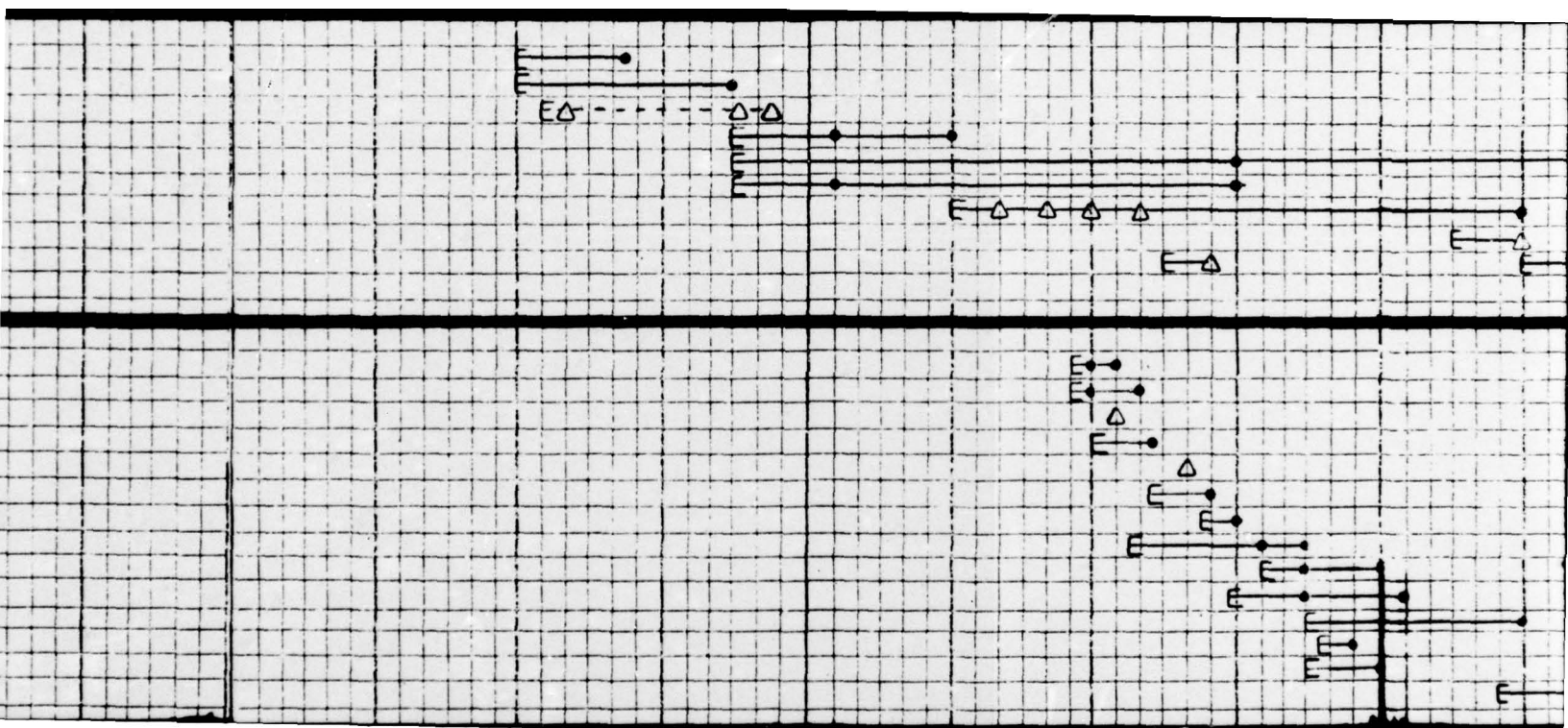
<u>Program</u>	<u>Labor in Man Months</u>
1	630
2	2960
3	430
4	115

Together these represent a picture of the variation in the effort required for the various programs.

1.11 Program Personnel

Persons entering into the manning for each program's development, possessed various background depending on their positions, education and experience. Personnel added to a program generally is an advantage to its development and personnel leaving prematurely is a disadvantage. There was no specific information provided on the availability of programmers. The results of a survey of the manning on the four programs is given in table 1-3.

LINE		FY 1969	FY 1970	FY 1971	FY 1972	FY 1973
		CY 1969	CY 1970	CY 1971	CY 1972	CY 1973
	J F M A M J J A S O N D J F M A M J J A S O N D J F M A M J J A S O N D J F M A M J J A S O N D J F M A M J J A S O N D J					
PROGRAM 1						
1 Certified Operating Sys.						
2 Certified Navigation Program						
3 Navigation Program Delivery						
4 Update/Certify Operating Sys.						
5 Update/Certify Operating Sys.						
6 Update/Certify Operating Sys.						
PROGRAM 2						
8 Interface Design Specification						
9 Program Performance Specification						
10 Program Design Review						
11 Program Generation						
12 Program Design Specifications						
13 Critical Design Review						
14 System Integration						
15 Operational Readiness Demonstration						
16 Final Op. Certification						
17 Final Op. Installation						
PROGRAM 3						
20 Mngt., Perf., QA, CM Plans						
21 Program Performance Specification						
22 Program Design Review						
23 Program Design Specification						
24 Critical Design Review						
25 Level 1 CPP and CPOM						
26 Level 2 CPP and CPOM						
27 Computer Program Test Plan						
28 Computer Programs Test Procedure						
29 Computer Program Package						
30 Integration Support						
31 CPOM						
32 Program Acceptance Test						
33 Final Operational Inst & Cert						
PROGRAM 4						
37 Computer Program Design Specifications						
38 Level 1 CPP						
39 Level 2 CPP						
40 Computer Program Test Plan						
41 Computer Program Test Proc.						
42 Computer Program Package						
43 Computer Program Operators Manual						
44 Computer Subprogram Design Doc.						



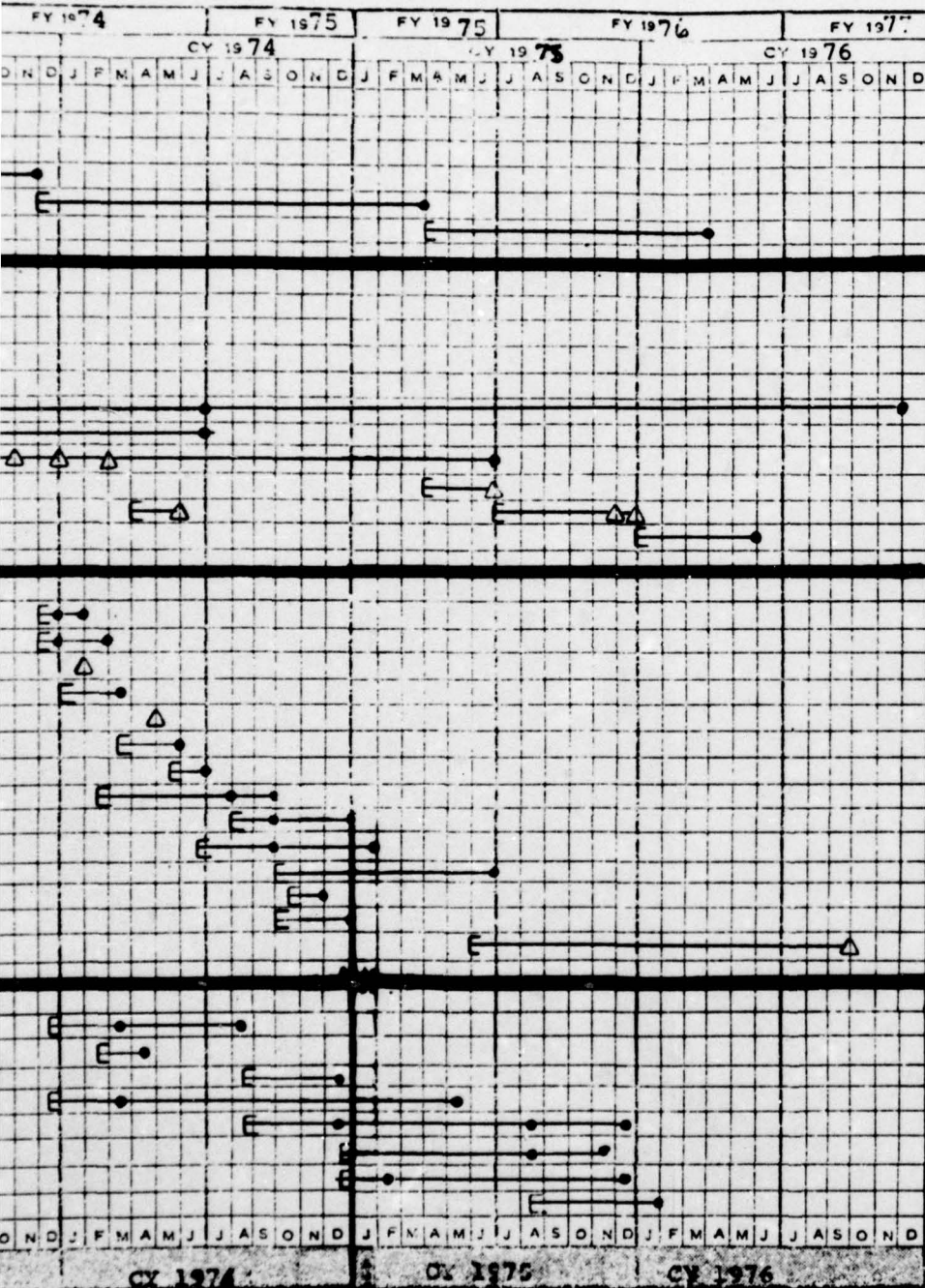


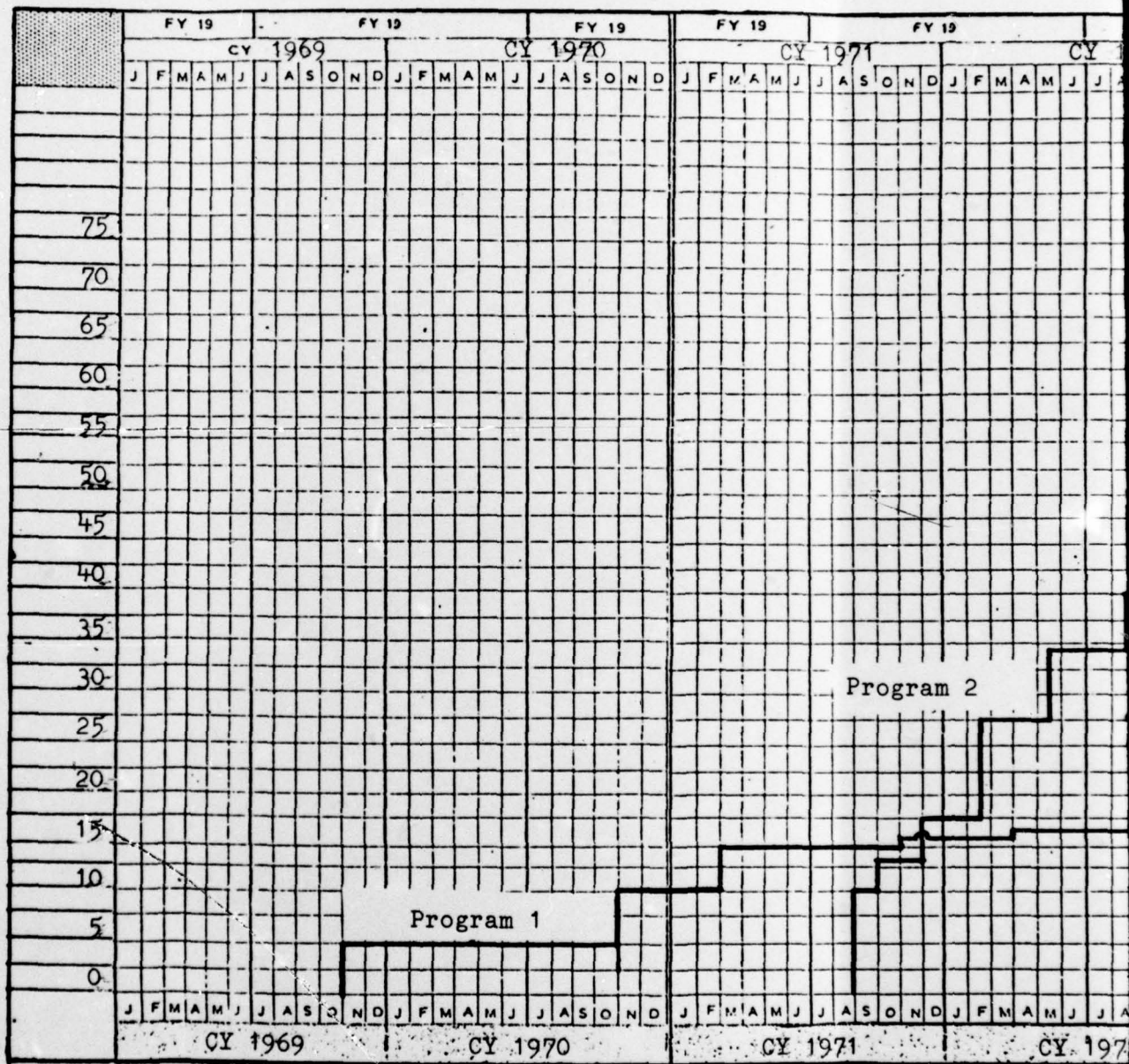
Figure 1-1
Major Contract
Milestone Schedule

Definitions:

- [- Start of item of work
- Δ - meeting date
- - Completion of item of work

3

ENGINEER MANNING ON PROGRAMS



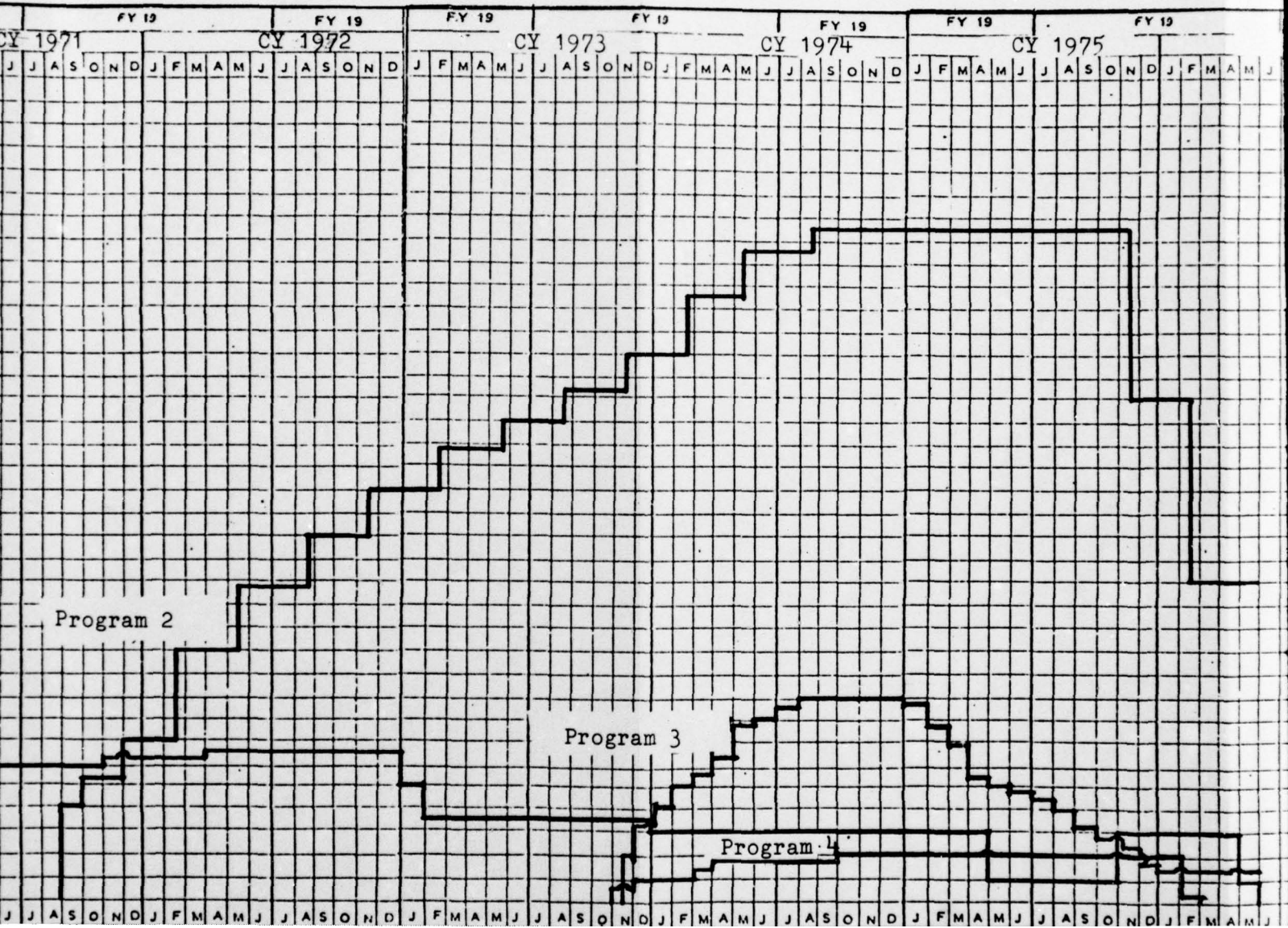


Table 1-3. Program Manning

Position	Number Persons Involved	When Joining Project, number persons															Number Persons Leaving Pre- maturely
		Highest Level		Formal Education		Years Experience Related to Job											
		BA	Graduate	0	1	2	3	4	5	6	7	8	8+				
		H.S.	BS	1 yr	2 yr	M	D	r	r								
PROGRAM 1																	
1. Manager	2			2												2	1
2. Supervi- sors	2			1	1											2	1
3. Clerical Typists	2	2						1								1	1
4. Sr./Pr. Soft. Eng.	0																0
5. Software Eng.	4		4							2	2						0
6. Associate Soft. Eng.	8		8					8									0
7. Soft. Eng. Aid	0																0
Trainee	0																0
PROGRAM 2																	
1. Manager	2		2													2	1
2. Supervi- sors	8		8							8							0
3. Clerical Typists	8	8								8							0
4. Sr./Pr. Soft. Eng.	9		9														0
5. Software Eng.	17		17									17					0
6. Associate Soft. Eng.	22		21					1		22							0
7. Soft. Eng. Aid	4	4						4									0
8. Trainee	5		5					5									0
PROGRAM 3																	
1. Manager	1		1													1	0
2. Supervisors	2		2													2	0
3. Clerical Typists	1	1								1							0
4. Sr./Pr. Soft Eng.	5		5													5	0
5. Software Eng.	2		2								2						0
6. Associate Soft. Eng.	4		4							4							0
7. Soft. Eng. Aid	1	1														1	0
8. Trainee	0																0
PROGRAM 4																	
1. Manager	2		2													2	0
2. Supervisors	1							1								1	0
3. Clerical Typists	1	1								1							0
4. Sr./Pr. Soft Eng.	3		2					1			1					2	0
5. Software Eng.	0																0
6. Associate Soft. Eng.	1		1							1							0
7. Soft. Eng. Aid	0																0
8. Trainee	0																0

1.11 (continued)

To compare this data between the programs, a personnel profile evaluation has been calculated as follows:

$$\text{Profile} = \sum_{1}^{(A-B)} (A-B) \times C \times (D+E)$$

where:

- A-B = number of program personnel
- A = number persons on project
- B = number persons leaving prematurely
- C = position weight factor
- D = number years experience
- E = number years of equivalent experience shown by formal education

and

Profile

$$\text{Program Person} = (\text{Profile}) + (A-B)$$

Position weight factor is taken from:

<u>Position</u>	<u>Weight Factor</u>
Manager	10
Supervisor	9
Clerical Typist	2
Sr/Pr. Soft. Eng.	7
Soft. Eng.	6
Assoc. Soft. Eng.	5
Soft. Eng. Aid	3
Trainee	2

1.11 (continued)

Equivalent years of experience is taken from:

<u>Highest Education Level</u>	<u>Equivalent Yrs. Exp.</u>
H.S	0.5
B.A., B.S., A.A	2.5
1 yr. grad.	3.0
2 yr. grad.	3.5
Masters	4.0
Doctors	6.0
+ 8 yrs. exp.	10.0

Result:

	<u>Average Profile</u>
Program 1 profile = 584.25	$\frac{\text{Profile, } 584.25}{\text{Persons, } 18} = 39.0$
Program 2 profile = 2747	$\frac{\text{Profile, } 2747}{\text{Persons, } 74} = 37.1$
Program 3 profile = 1004	$\frac{\text{Profile, } 1004}{\text{Persons, } 16} = 62.8$
Program 4 profile = 659	$\frac{\text{Profile, } 659}{\text{Persons, } 8} = 82.4$

This evaluation is based upon the position and years of experience of personnel at the beginning of the project and does not account for the years of advancement and experience while on the project. It also illustrates a result of a hiring freeze that was in force when Program 3 and 4 began.

The result shows a trend toward a greater percentage of manning by management on small programs.

SECTION 2

TOP DOWN PROGRAM DEVELOPMENT (TASK 2.1)

2.0 INTRODUCTION

Top down program development comprised a reasonably simple set of concepts which had natural appeal due to the many benefits afforded to the management and development of a software project. Top down program development included program design, production (coding and checkout) and formal testing. Practical utilization of these ideas in a serious, large-scale program development called for the preparation of methods, procedures and tools, some adaptation to the particular software architecture, and consideration of management and contractual requirements. This section describes an actual application of top down program development which has proven successful on Programs 3 and 4 by allowing completion of these programs within cost and on schedule. With variations on the techniques used, the management, design, production and testing proven on these programs is applicable to a wide range of software developments.

The principle advantages of top down program development compared to other development methods were:

- a) The ability to establish multiple fixed and visible milestones within the production and testing of a computer program.
- b) Early production and verification of the control logic which enabled a program to execute as a component of a data processing system.
- c) The ability to begin system integration prior to the completion of program development.

2.1 METHODOLOGY FOR TOP DOWN PROGRAM DEVELOPMENT

The program development method used on Programs 3 and 4 consisted of four phases of work; analysis (what), design (how), production, and testing.

During the first phase (program analysis), system requirements were translated into program requirements. What the program was to do was established in functional terms. Data processing equipments were identified and their interfaces to software were preliminarily defined. The overall software architecture, dependent upon the program executive used, was established. The allocation of functions to software subprograms and interface of subprograms to the executive and between subprograms was in the formative state. Communication within the design group and management was good. The results of the analysis phase were visible in the form of a program performance specification. There was seldom any problem with visibility over progress and status because the analysis phase was early and short, functions of the program were being emphasized, and design group and management communication was good.

The second phase (program design) pertained to how the functional requirements were to be implemented in a computer program. The objective was completion of the program design and generation of the Preliminary Program Design Specification. But in addition, program design for top down development of the example programs included a breakdown of all code to be generated into identified logical design elements. These elements were organized in a tiered hierarchy which resembled an inverted logic tree. The uppermost element contained the highest order of control logic of the program. The elements to which the uppermost element passed program CPU control were situated beneath it at the second tier of segmentation. In performing their functions, the second tier elements could in turn pass control to other elements which

2.1 (continued)

were situated in lower tiers. In this organization, the greater the number of the tier, the lower its position in the hierarchy. Top down design continued this process downward to as many tiers of program elements as were required to satisfy the functional requirements of the program. The lowest tiered elements performed routine algorithmic and data processing functions directly related to the purpose of the program, many were called by more than one higher tiered element. The assignment of elements to tiers met the following rule: An element may call upon only lower-tiered elements, and an element must return control to the calling element.

Program production and testing proceeded element by element from the top tier downward. After the proper operation of each element was verified, that element was integrated into the subprogram; which, when completed, became a part of the deliverable program. During the software system development, the program elements which had not yet been produced and integrated into the system were represented by "stub". A stub was a dummy program element which took the place of a program element not yet coded or checked out. Stubs were placed on the source library just as real program elements were. In the object program the stubs satisfied the element calling (control passing) functions depicted in the tiered hierarchy. A stub also modeled the consumption of computer resources through the occupation of main memory and processor time use. A stub contained as many of the actual element characteristics, or simulations of characteristics, as a programmer or system designer deemed necessary for the particular program development.

The stubs defined the element name and provided as a header commentary (a required convention) a description of the inputs, processing function and outputs so the listing could be read as

2.1 (continued)

a complete representative of the program design. Throughout program production the combination of completed elements and stubs provided a complete representation (target machine model) of the operating program modules.

The segmentation of the program design into discrete and well-identified program elements benefited management visibility both during and after production. The function allocation at the element level was simple. These program elements were generally less than one hundred memory words long, aiding visualization of design. Top down design also localized related functions, rather than scattering them through large portions of a program. These program elements were to program design as printed circuit cards were to electronic hardware design. The logical and discrete division of the program made the program easier to understand, learn, review, modify and maintain. Program changes were isolated and localized, resulting in greater ease of preparation of change data and supporting rationale. These design properties were not unique to top down design, top down disciplines tended to yield a more definitized design in the code and did so uniformly throughout the entire program.

Management visibility and control was also enhanced by the subdivision of the long production period into several shorter periods marked by significant intermediate milestones. Top down design made this subdivision possible. These intermediate milestones were accomplished through the production and testing of logically selected sets of the program elements.

One of the many benefits to be gained through top down software development was the early realization of the proper operation of the program's control functions (e.g., it would start up, cycle, and interface properly). This avoided the production difficulties created by a disproportionate attention given to the application (purpose oriented) functions prior to the satisfactory

2.1 (continued)

working of the control (EDP system oriented) functions.

After individual program elements had been coded and checkout tested, they were integrated into a software system. The produced elements, with the required stubs, were then tested as a whole program without affecting the production of lower tiered elements. Detailed design of lower tiered elements may have occurred during or after this portion of program testing. The practice of design-a-bit produce a-bit was not considered satisfactory because the produced software was required to pass through comprehensive and rigorous design reviews prior to its production. Also, the program was designed as an integral whole to better attain efficient use of computer main memory and processing time. This method of modeling the final program design by accurately stubbing the uncoded elements of a program facilitated early appraisal of the developing program.

2.2 PROGRAMS 3 AND 4 TOP DOWN DEVELOPMENT

The methods of top down software development were applied to the development of two shipboard command and control system computer programs, Programs 3 and 4.

Program 3 comprised a set of system specialized functions which supplemented a furnished baseline real-time operating system to form the complete software operating system. The finished size of Program 3 was 26,600 32-bit words.

Program 4, a real-time application program, was the software component of a complex process control system. Its finished size was 13,150 32-bit words.

2.2 (continued)

Figures 2-1 and 2-2 illustrate the development schedules for Programs 3 and 4. Each program was developed in the four distinct phases of work; analysis, design, production and testing.

Work on Program 3 began in early November, 1973; while that on Program 4 began in late December, 1973. Contracts required that each program use top down development methods that would allow incremental program deliveries to be made during the production phases. The functional content of each incremental delivery was established by the customer, but the top down development methods were left entirely up to Sperry Univac. The methods employed were the result of an earlier top down concept which began formulation in 1972. Compliance with a furnished set of programming (design and coding) standards and conventions was also required. Adherence to the use of structured constructs was among these standards. These developments were also aided by a number of software development tools, hosted on a large-scale time-sharing interactive commercial system (see sections 3 and 9). These tools were accessible through remote alphanumeric terminal devices.

In each case the developments began with the preparation of a detailed project plan containing the following:

- a) Definitive list of the items of work.
- b) Detailed planning schedules.
- c) Itemized list of deliverables.
- d) Organizational responsibilities and key personnel.
- e) Plans for reporting to the customer and within the company.
- f) Financial (budgetary) planning.

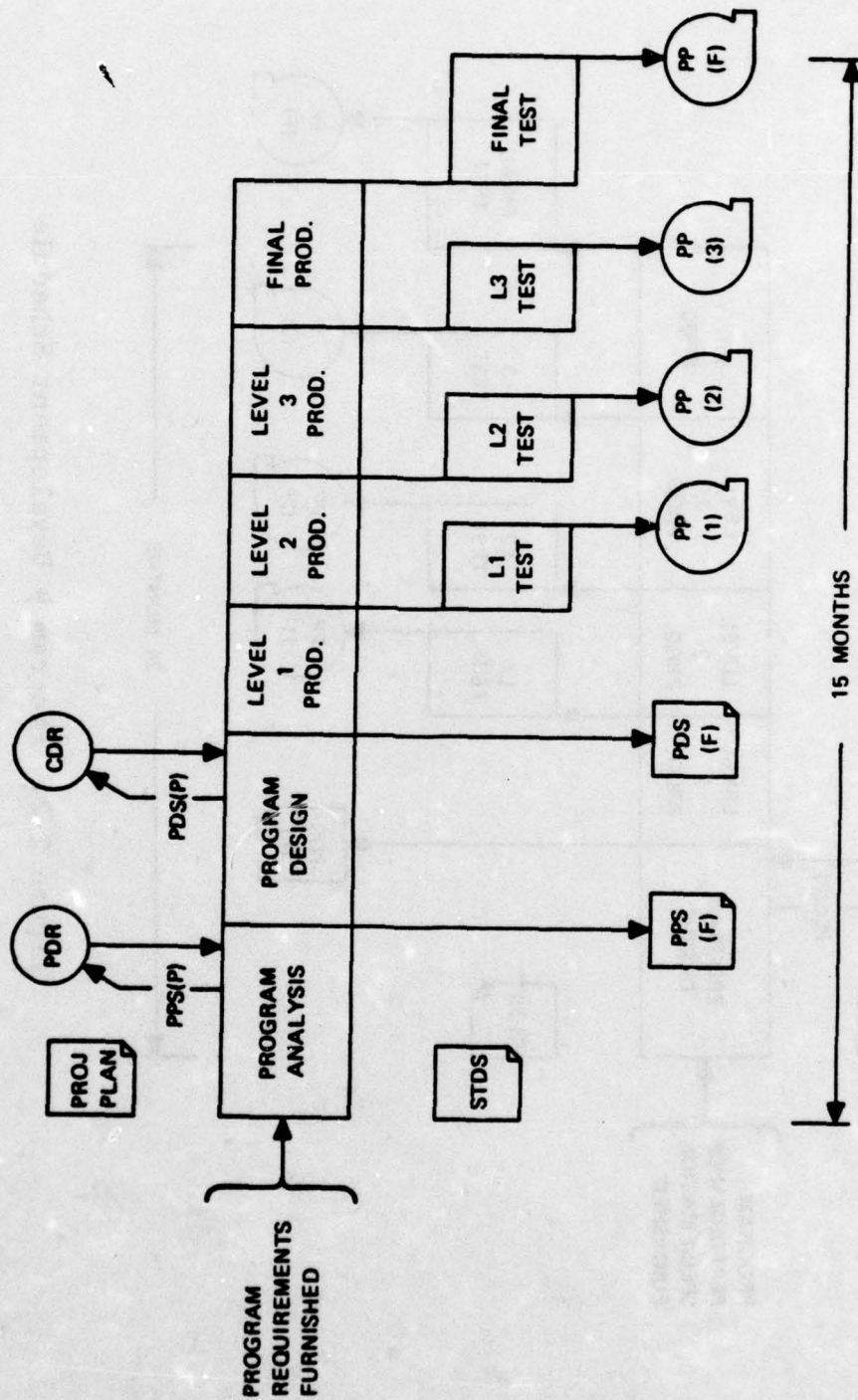


Figure 2-1. Program 3 Development Schedule

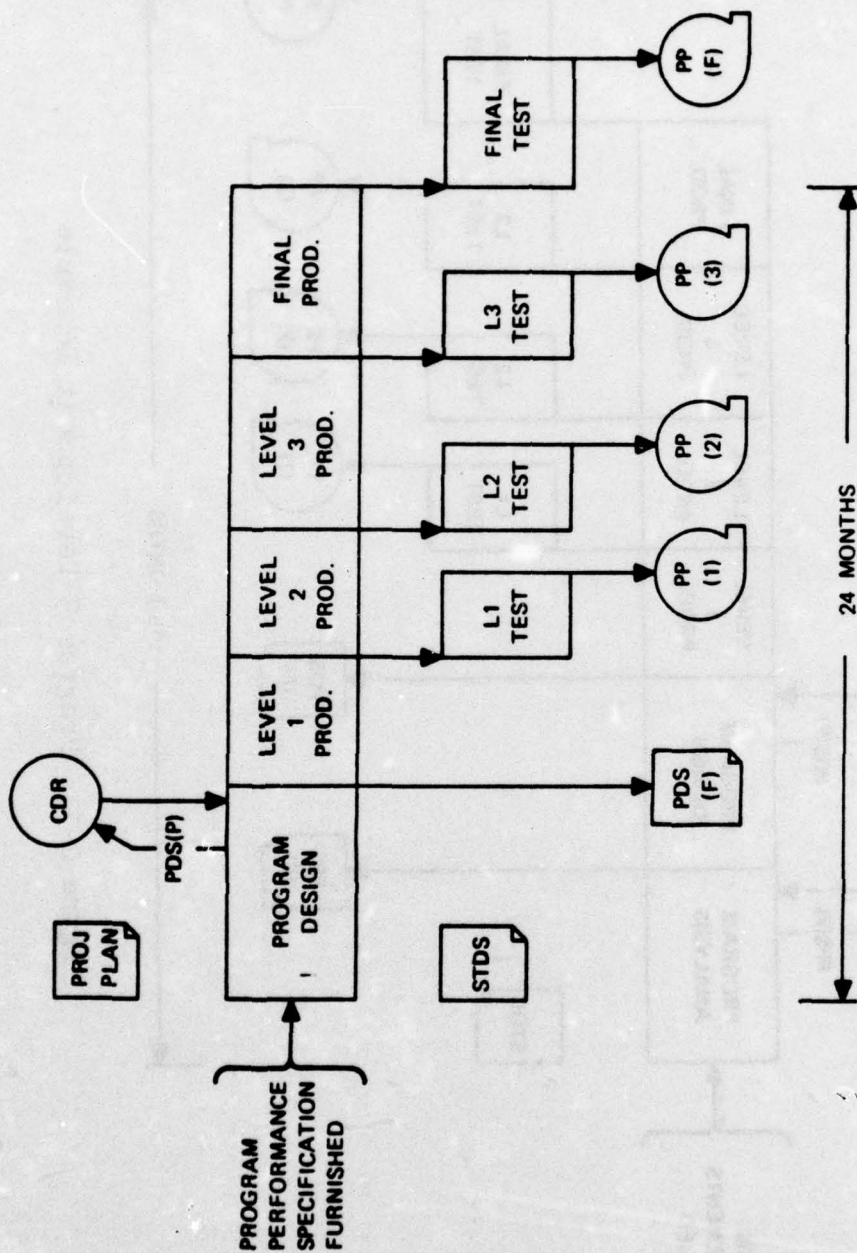


Figure 2-2. Program 4 Development Schedule

2.2 (continued)

- g) General description of the development methods and facilities to be used.
- h) Description of the software quality assurance, testing and software configuration management to be performed.
- i) Support software required.
- j) Project risk analysis.
- k) A number of work procedures were also provided to development personnel.

The initial product of the program analysis phase was a preliminary program performance specification (PPS(P)). The PPS stated in functional detail what the program was to do. This specification was comprehensively reviewed and commented upon by the customer and his consultants in a process known as the preliminary design review (PDR). The resolution of these comments was incorporated into the PPS and a final (PPS(F)) was published and delivered. This document was the functional baseline of the program to be designed. (Note on figure 2 that this document was a customer furnished item to the development of Program 4.) The contractor development of Program 4 began with analysis proceeding the design phase.

The initial product of the program design phase was a preliminary program design specification (PDS(P)). This specification was also reviewed and commented upon in a process known as the critical design review (CDR). The resolution of these comments accordingly resulted in the publication and delivery of a final design specification, the PDS(F). This document was then the design baseline of the program to be produced.

2.2 (continued)

Top down program production and testing followed. The production and testing were performed incrementally in four levels, including the final. The output at each level was a deliverable program package (PP) consisting of source and object records and listings.

This level production is described later in greater detail.

Parallel efforts were performed by other functional organizations to supplement the work of the development group. Real-time wrap-around simulation software, produced to support the latter stages of checkout testing, became an integral part of the formal test system. The formal testing and software quality assurance monitoring and auditing were performed by another group of personnel. The preparation of the test system included the development of deliverable test plan and test procedure documents, the development of specialized test tool software and the incorporation of the real-time wrap-around simulation. The objective of the testing was to assure compliance to the PPS. Goals of the test system development included test automation, repeatability and test system viability. Other external organizational functions provided were data management and software configuration management. (See sections 4, 5, and 6).

2.3 TOP DOWN DEVELOPMENT TECHNIQUES

Specific top down development techniques were established by Sperry Univac to meet the special constraints and requirements of the systems' software environment. These techniques consisted of:

- a) Dividing the program into large program modules in accordance to the design requirements of the real-time operating system software.

2.3 (continued)

- b) Beginning each modules hierarchical structure with the real-time executive.
- c) Allowing for a natural element breakdown of each module using the coding structures of the required high level language, CMS-2.
- d) Designing each program module as a structural hierarchy of elements called design objects.
- e) Adhering to a set of rules for determining the hierarchy of design objects and accepted method of coding design objects.
- f) Providing control and visibility of the code production by the status of each design object and plan for its production.
- g) Producing and delivering the program in four levels of completeness such that each level represented a full scale executable model of the final program.
- h) Developing the qualification testing for each produced level of completeness and qualifying each level of the operational program as a baseline for the next level.
- i) Providing for early integration of the full model operational program after delivery of level program.

The application of these techniques allowed the design quality and the production visibility and control benefits of top down development to be amply realized.

2.3.1 Executive Program Constraints

Programs 3 and 4 were executed under control of a furnished kernel executive. This executive established definite software architectural constraints upon the application of the top down development concept. The executive required that each program be divided into large program modules; and that each module receive CPU control through a predefined set of entrances from the executive. No single element was to be placed at the apex of the top down hierarchy, but several modules and several entrances within each module formed the first apparent tier of program elements. Each module was strictly partitioned from the others, allowing the modules to be developed independently. Each program module could be separately designed and produced by top down techniques. This did not eliminate accounting for necessary interfaces with other modules and with the executive, but each module could be structurally developed in a manner which ignored the parallel development of the others. Each module had several entrances which formed its top tier of program elements. If the program logic for each entrance had been separately developed using top down techniques, many elements common to two or more entrances of the same module would have been redundantly produced. To avoid this, the executive was considered as the unseen apex of the top down hierarchy, although the first tier of a module was shown with the several entrance elements. Instead of a single hierarchy as seen in diagram a) of figure 2-3, there resulted many hierarchies as in diagram b) of figure 2-3, with each being subordinate to the executive.

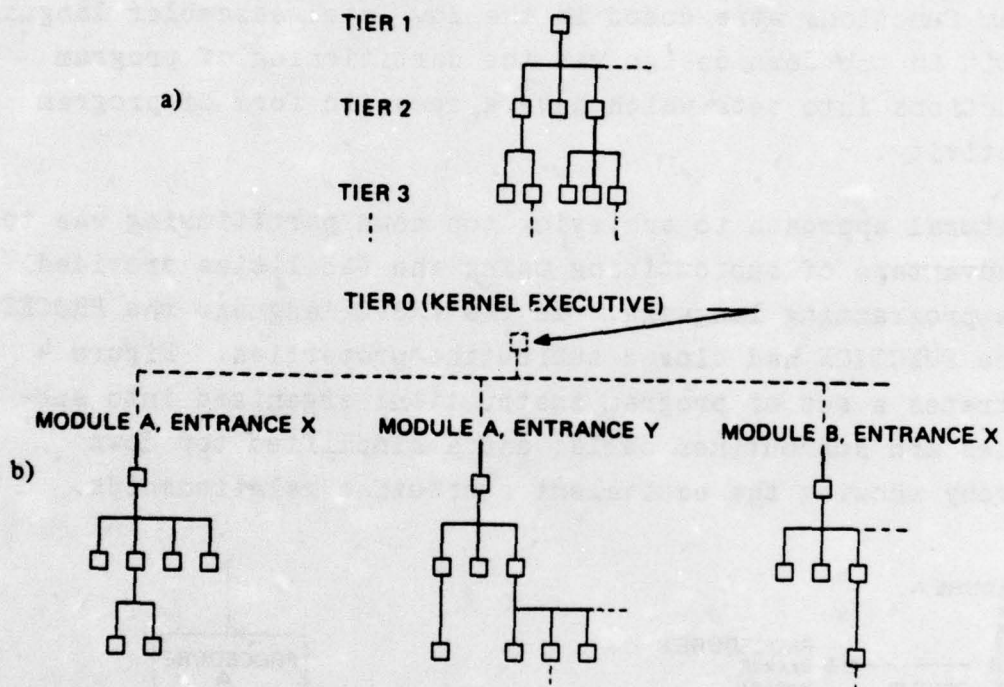


Figure 2-3. Top Down Hierarchies

2.3.2 Program Language Elements

The two programs were coded in the Compiler Monitor System (CMS-2) language. CMS-2, developed for use with the AN/UYK-7 computer, was comprised of a compiler, assembler, librarian, monitor, program loader, debugging tools, and utility functions. The high level compiler language was preferred, but a few of the program functions were coded in the low level assembler language. Implicit in top down design was the partitioning of program instructions into sets which have a specific form of program connectivity.

One natural approach to achieving top down partitioning was to take advantage of subroutining using the facilities provided by the programming language. In the CMS-2 language the PROCEDURE and the FUNCTION had closed subroutine properties. Figure 4 illustrates a set of program instructions organized into subroutines and subroutines calls, and a simplified top down hierarchy showing the equivalent subroutine relationships.

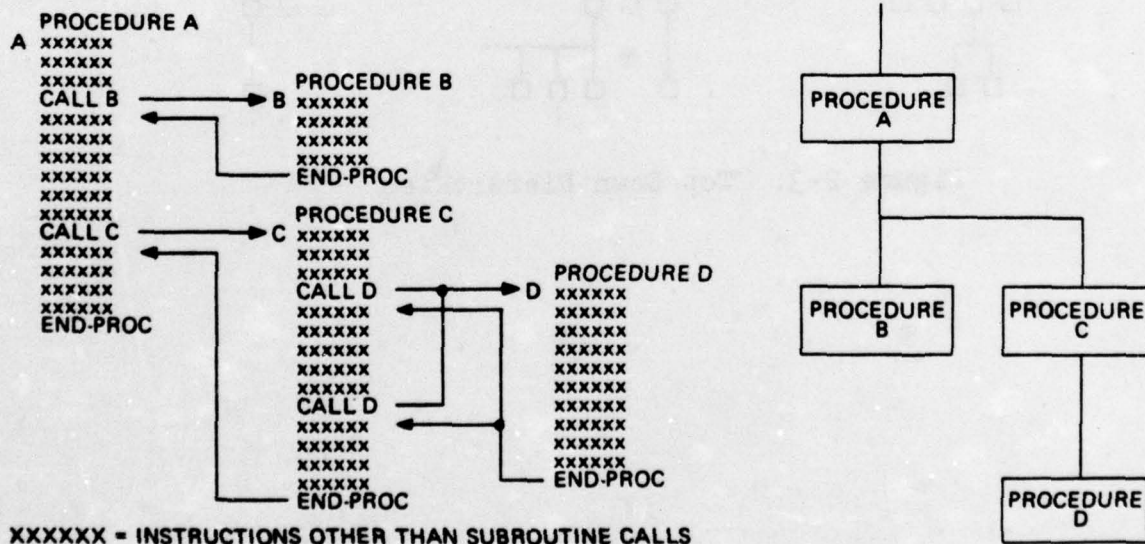


Figure 2-4. Subroutining in a Top Down Hierarchy

2.3.2 (continued)

Data elements were also included in the top down hierarchy. The CMS-2 language elements containing data were system data designs (SYS-DD) and local data designs (LOC-DD). Each of the four language elements (PROCEDURE, FUNCTION, SYS-DD, and LOC-DD), or a closely knit group of them, were then qualified to be represented by an element block on a top down hierarchy.

2.3.3 Top Down Documentation

Several documentation conventions were required to document the top down design, and a program was used to automatically generate design object drawings and tabular data. Each of the program elements on the hierarchies were termed design objects. The top down hierarchies were termed design objects. The top down hierarchy drawings were called design object drawings. The symbol shown in figure 2-5 was used to represent each design object.

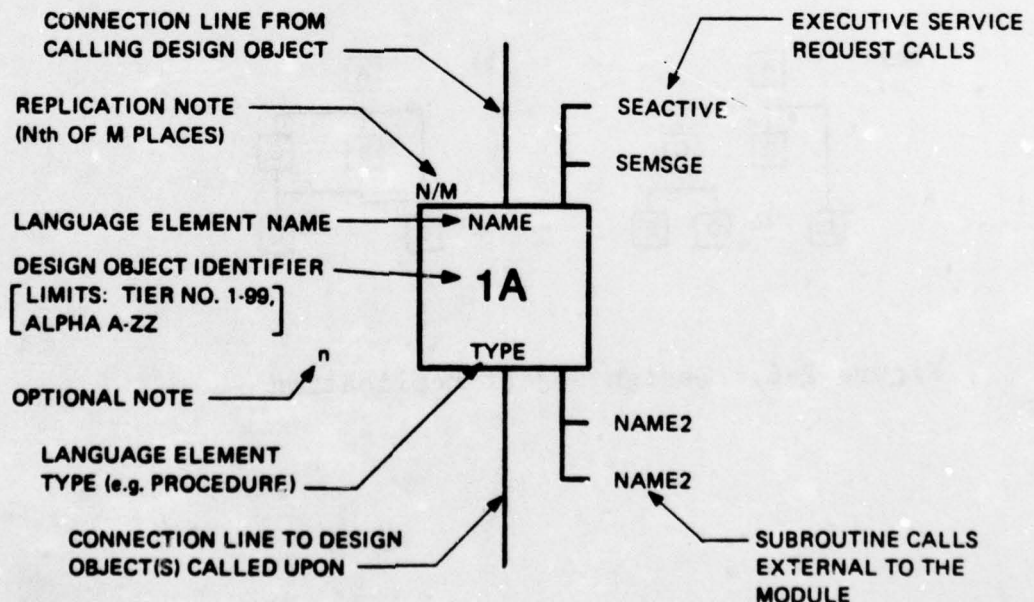


Figure 2-5. Design Object Symbol

2.3.3 (continued)

Each design object was given a two to four character identification code for reference purposes and was further identified by language element name and type. The numeric part of the design object identifier code specified the tier to which the design object belonged. The alphabetic part of the identifier was arbitrarily chosen to uniquely identify the design object within that module. In the design object drawings, the boxes represented the design objects of a module and the connecting lines represented calling and return path linkages. A design object called upon by N other design objects would appear on the drawings N times, thus simplifying the preparation of the drawings by eliminating nodes or convergences within the design object hierarchy. A resulting design object drawing looked as shown in diagram a) in figure 2-6 rather than as in diagram b).

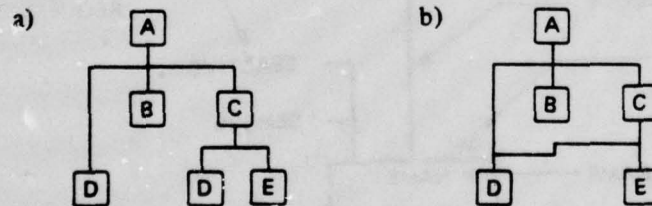


Figure 2-6. Design Object Replication

2.3.3 (continued)

Occasionally, this redundant illustration of design objects greatly increased the size of the design object drawing. This occurred when a design object was called a large number of times, and that design object then called upon a number of others.

When the number of drawing sheets was considered excessive, the situation was remedied by detaching the repetitious portion of the hierarchy from the main drawing, providing it on a separate drawing, and showing only the calls to the single upper-most design object with a noted reference to the separate drawing. See figure 2-7.

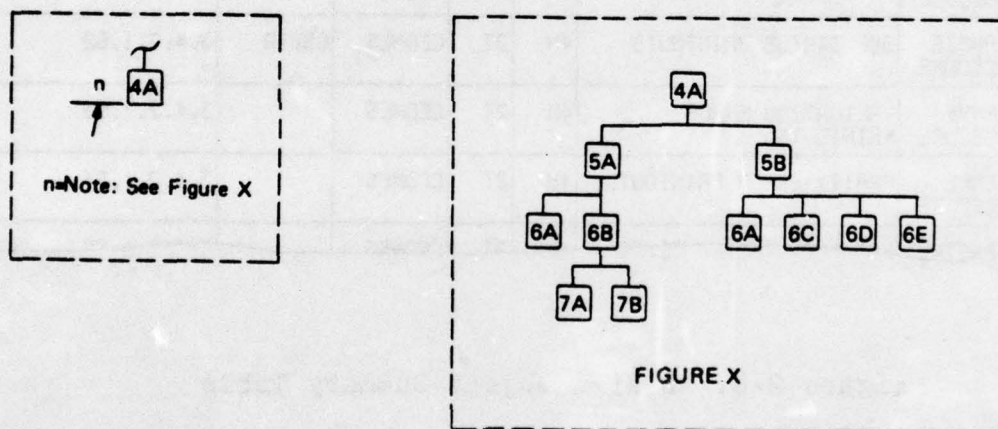


Figure 2-7. Detached Hierarchy

2.3.3 (continued)

A design object summary table was provided with each design object drawing. This table summarized the information on the design object drawings and provided a brief design object description and a reference to the detailed description paragraph for the design object in the program design specification. The design objects were included in the table by identification code in alphanumeric sequence. See figure 2-8.

D.O. ID	LANGUAGE ELEMENT	D.O. DESCRIPTION	CALLS TO FROM		ESRS	EXTERNAL CALLS	CPDS DESCRIPTION PARAGRAPH
3AE	EDCPNSEL PROCEDURE	SDC SELECTION PRINTOUTS	4H	2T	CEDMES		3.4.2.1.61
3AF	EDCPNSTS PROCEDURE	SDC STATUS PRINTOUTS	4H	2T	CEDMES	CSBTA	3.4.2.1.62
3AG	EDCPNVE PROCEDURE	VALIDATION ERROR PRINTOUTS	4H	2T	CEDMES		3.4.2.1.63
3AH	EDCPNPE PROCEDURE	PARITY ERROR PRINTOUTS	4H	2T	CEDMES		3.4.2.1.64
3AI	EDCPNETF	SDC STATUS PRINTOUTS	4H	2T	CEDMES		3.4.2.1.65

Figure 2-8. Design Object Summary Table

2.3.3 (continued)

The data required for automatic generation of the Design Object Networks and Design Object Summary Tables were maintained as card images in the computer files from which the drawings were automatically produced by a software tool on the Univac 1108 time-sharing system. Changes or corrections to a drawing were easily and cheaply made by a simple revision of the drawing file.

Figure 2-9 shows the computer drawing for a small program module of approximately 1000 memory words. Notice how the module entrances are signified. Note also the replication of design object 3A.

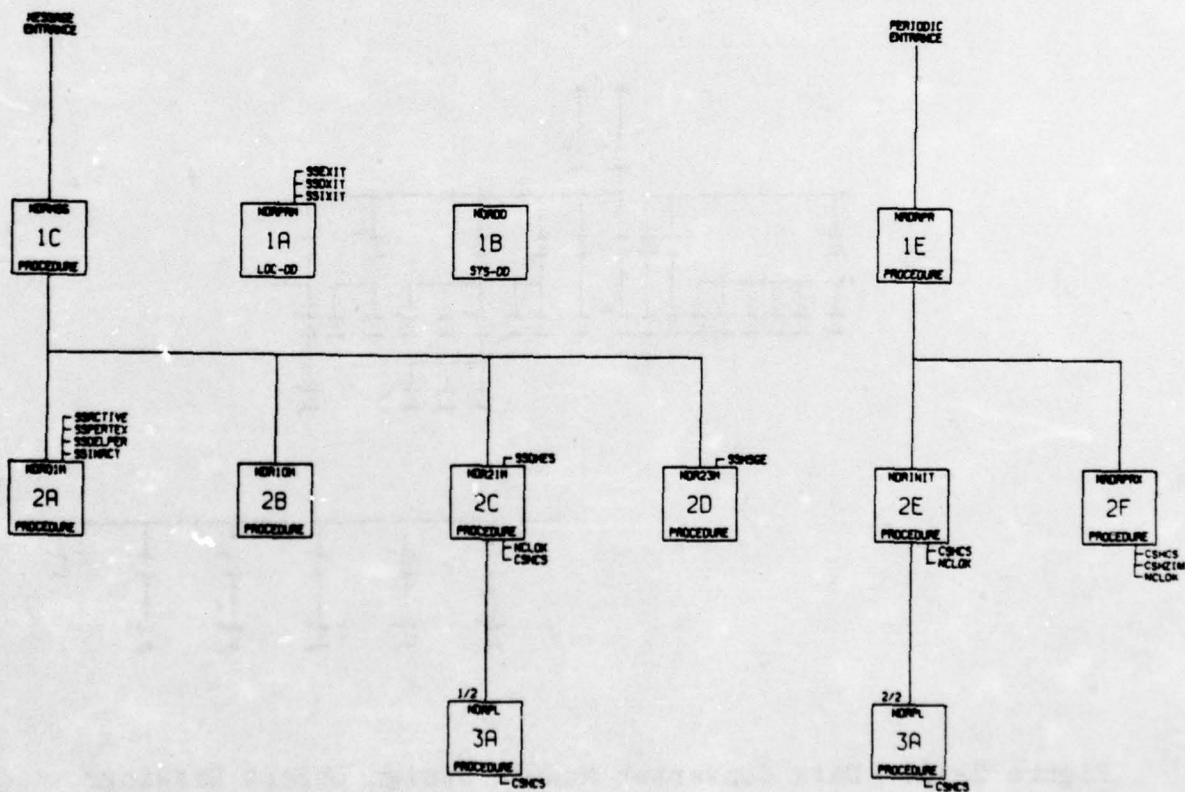


Figure 2-9. Dead Reckoning Module Design Object Drawings

2.3.3 (continued)

Figure 2-10 depicts the three sheets of design object drawings for a large module of about 3000 memory words.

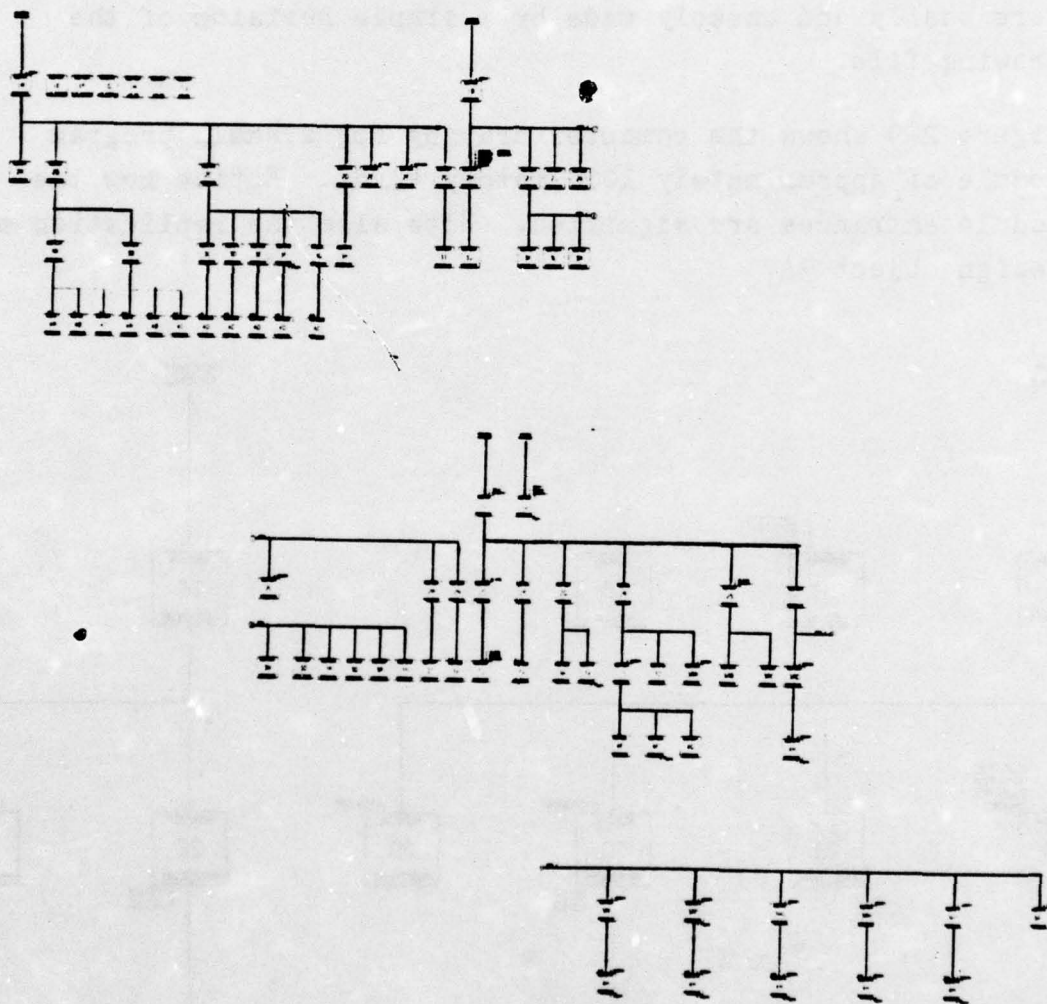


Figure 2-10. Data Converter Module Design Object Drawings

2.3.3 (continued)

These drawings and tables became key parts of the program design specifications.

2.3.4 Top Down Rules

The following rules were applied to the top down development of Programs 3 and 4:

- a) Each program module was individually top down developed.
- b) Each line of program code was defined within one and only one design object.
- c) All design objects called by another design object returned to the calling design object.
- d) Tier 1 contained the entrance procedures which were accessed by the kernel executive and exit from the module back to the executive were from these same design objects rather than from other design objects within the module.
- e) Data design objects were depicted in tier 1 (without program control connection lines).
- f) Input/output chains were preferred at tier 1 in data design objects.
- g) A design object called only on lower tiered design objects.
- h) If more than one design object called a subroutine, that subroutine was depicted as a design object. However, if only one design object called a subroutine, that subroutine was depicted and documented as either a unique design object or as part of the calling design object.
- i) Each stub was coded to contain all Executive Service Request calls and all design object calls that were intended for the design object.
- j) Each stub was coded to model the represented design objects in the consumption of computer memory and processor time.

2.3.5 Top Down Production

Upon approval of the program design, program production began. Top down production of the program meant that it was coded and debugged in discrete incremental steps called levels. To define a level for a given program module, a set of the module design objects was selected for production. The selection was based on objectives of program operation and test as assigned to the level. The completion of the coding and checkout testing of this set of design objects, together with the stubs of the other design objects, represented a completed level of the program module under development. This program module was then available for the overall program integration and qualification testing which may have occurred concurrently with the production of the next level. Complete implementation of function could occur at any desired programmed level. Normally, the design objects selected for completion at a level were selected from the top of the design hierarchy down, such that any stubbed design object would have only completed design objects above it. At each completed level, the existent completed design objects and the remaining stubs provided a target machine model of the complete program. The stubs consumed computer resources by reserving computer main memory and by loop cycling an estimated time to represent processor usage.

Programs 3 and 4 were produced in four levels. The general objectives of the first level were to produce each module's entrance procedures, its data design objects and the procedures performing initialization functions (real-time initialization or startup). All design objects not produced in real and complete code as a part of the first level were stubbed. All Executive Service Requests calls were coded regardless of whether they occurred in the completed or stubbed design objects. Thus, with the completion of the first tier entrance procedures, and the coding of the Executive Service Requests Calls, all executive interface code was in place at the completion of the first level.

2.3.5 (continued)

The general objectives of the second level were to complete design objects below the first tier entrance procedures which performed additional task control functions, and those design objects which provided or obtained data external to the module (input/output and intermodule communications). For many program modules the objective of the third level was to fully implement some of its major functions. This meant that hierarchies were completed vertically in selected localities. At the completion of the fourth level of production, all coding and checkout testing was done and the program was ready for final qualification testing. During each level of development, coding and checkout testing were monitored and the condition of each design object was recorded on the status reports. Therefore, not only was the use of the design object a means of documenting the top down design of each program module, it served as a means to manage the top down code production and provided the customer good visibility of production status.

As part of the design phase, the following data was prepared by the responsible programmer for each design object:

- a) Estimated size in memory words.
- b) Estimated average execution time.
- c) Required executive service requests.
- d) Required external calls (e.g. mathematical subroutines).

The compilation of estimated size, for example, gave a total estimate for the size of each module, since each estimate was based on fairly detailed requirements, the total estimated size was found to be quite close to the resulting actual size. A production status report (see figure 2-11) was prepared and maintained weekly for each module showing the status of each design object in the following increments:

PRODUCTION STATUS REPORT

Program SP

Date 30 Sep 74

Sub-program DC

Level 2

Sheet 1 of 3

Design Object		Size in Words E=Estm. A=Actual	Status							Remarks
ID	Description		Stub	Code	Desk	Unit	Module	In- tegration	Check- out	
1A	Priority Entrance	42 A							X	
1B	Message Entrance	34 A							X	
1C	Periodic Entrance	35 A							X	
1D	Demand Entrance	21 A							X	
1E	SDC IOC Interrog Chains	118 A							X	
1F	EAT IOC Chain	25 E	X							Rqmts TBD
1G	Term. Active IOC Chains	6 A							X	
1H	Preamble	20 A							X	
1I	IBRAT	20 A							X	
1J	Data Tables	604 A							X	
2A	Data Time Tag	25 E	X	X	X					
2B	Data Validation	50 E	X	X	X					
2C	Format Abrogation Data	10 E	X	X	X					
2D	Process Description Data	40 E	X	X	X					
2E	Update Companion Data Base	15 E	X	X						
2F	Ant. F. Ship Control of Update	10 E	X	X						
2G	Check EAT Results	60 E	X							Rqmts TBD
2H	Preset Message	79 A							X	
2I	Operator Requests	25 E	X							
2J	Data Node Registration	25 E	X	X						
2K	SDC Data Update	25 E	X	X						
2L	Display Requests DDRT	15 E	X	X						
2M	DDRT Priority Error	15 E	X							Not scheduled for Level 2
2N	EM Log Status	1 E	X							

Figure 2-11. Production Status Report Forms

BEST AVAILABLE COPY

2.3.5 (continued)

- a) Stubbed
- b) Coded.
- c) Desk debugged.
- d) Unit debugged.
- e) Integrated into module.
- f) Checkout testing complete.

The program production progress was monitored at least once a week by observing the status of these design objects required for the next level completion. Deviations from schedules or expected size were visible early in the production phase for management to take effective corrective steps. For example, corrective actions may have been assigning senior personnel to assist the programmer assigned to a module before that module development could fall behind schedule. Corrective action taken for deviations in expected size, for example, was the update of the size of the Production Status Report, such that the visibility of the total best estimated memory requirement was given. At the completion of each production level, the program was made available to quality assurance personnel for inspection and qualification testing.

2.3.6 Top Down Testing

One of the major advantages of top down software development was the incremental integration and testing of program levels prior to the completion of program production.

When each module of the program was completed to Level 1, it was integrated into a system and executed with the executive operating system software, a simulation program, and data extraction

2.3.6 (continued)

software. Proper execution of each module entrance was tested causing execution of both the completed and the stubbed design objects. Those functions which were complete at this level were evaluated in full and the control logic and connectivity of the integrated program was evaluated. When the qualification testing of this integrated set of Level 1 modules was completed, the resulting Level 1 program was established as a baseline and delivered. Concurrent with the qualification testing of the Level 1 baseline program, development began on the next baseline, the Level 2 program.

Generation of the test support software and documentation was also completed in a top down manner. Because testing of each level was based on the previously validated test, the tests beyond Level 1 were built upon the tests performed at the previous level.

The final level of testing was a complete functional test of the program's performance similar to that which would have been performed if the program had been produced in the traditional manner.

Because program performance requirements were assigned to specific design objects, and each design object was specifically documented as a unique program element, faults detected by qualifications tests were normally traceable to one specific segment of code. This gave a coherence to detected problems which minimized the time required for error isolation and correction. Needed corrections and changes to design were most often made in single design objects. This localization of affected code facilitated modifications, and helped to avoid partial (erroneously incomplete) corrections.

2.3.6 (continued)

Each program level was delivered following completion of testing. These deliveries facilitated system integration far in advance of the final completion of any of the many system programs.

2.4 SUMMARY

Both of these program development efforts were considered to have been highly successful (i.e. both programs were completed on time, within budget, and both programs consistently performed as required.) The combined effect of top down programming, new hosted development tools, new design constructs and language constructs has been an increase in program productivity. Top down programming is partially responsible for this improvement. Program 3 was developed at a cost in programmer man-hours of 87% of the formerly expected productivity for the development of real-time operating system software of this particular mix of complexity. Program 4 was developed at a cost in programmer man-hours of 84% of the formerly expected productivity for a real-time applications program developed from a furnished performance specification. These programs will continue to benefit from the application of top down concepts whenever maintenance or modification is performed.

During and since these two development projects the top down design documentation concepts and tools have been used to document, several programs. This form of design documentation has met with favored acceptance from both programmers and customers. In one case a specialized program was produced to automatically generate from existing source code the files from which the drawings are produced. This tool has a high potential value when applied as a quality assurance tool to verify the fidelity of the actual design hierarchy of a coded program against its specified design hierarchy which has been previously documented.

2.4 (continued)

A large software development project utilizing these top down methods is now underway. This project comprises a system of programs having a total size of many times that of the original two programs. Confidence in the ability to successfully complete this software has been considerably enhanced by the success of programs 3 and 4.

2.5 CONCLUSION

Top down development of computer programs is recognized as a significant advance in software development technology. Programs which are top down designed benefit in reliability, maintainability and adaptability to change from the good design practices which top down disciplines tend to force in a natural and uniform way. The logical and discrete segmenting of a program make it easier to understand, learn, review, modify, and maintain.

Production and testing by top down methods greatly improves the probability of completing a programming project within cost schedule; and, further, it benefits the overall system integration process. One of the greatest of the many benefits to be gained through top down production is the early realization of the proper operation of a program's control functions (i.e. it will start up, cycle and interface properly). The ability to incrementally produce and test a program permits the establishing of significant and tangible intermediate production milestones. Those responsible for production are able to recognize and isolate production troubles far more effectively than through traditional methods. Program modifications tend to be better isolated and localized, resulting in greater ease of preparation of change data and supporting rationale. With proper advance planning it is possible to begin system integration prior to the completion of program development.

2.5 (continued)

Graphic depiction of the top down design organization of a program aids the overall application of the top down concept, allows for control and visibility during production, and benefits continued maintenance.

Strict adherence to the top down rules and strict application of top down production is usually possible for simple programs free of sophisticated software architectural constraints. Some routines that require reentrant and recursive execution and require minimal execution time may require returns at any level. Also, for large integrated systems of programs, the original functional requirements may change after production is begun and may result in a redesign and alteration of some delivered software.

Top down program development can be applied to any case. In special routines a Top Down Rule can be waived and production plans can be adjusted whenever a new or changed software requirement is given. Also, as described in the application of Top Down Program Development on Programs 3 and 4, adaptation was made for the executive relationship to a program module and design accommodated the specified high level language.

The methods described were applied to producing Programs 3 and 4, and these programs were only two of seven programs developed in a Top Down manner by other agencies, then delivered in levels and integration to form the total program. Therefore, not only was integration of the total program far ahead of its final development, the process of integration and testing as a system of programs proceeded in much the same manner as that of the described integration and testing of Programs 3 and 4.

AD-A040 049

SPERRY UNIVAC ST PAUL MINN DEFENSE SYSTEMS DIV
MODERN PROGRAMMING PRACTICES STUDY REPORT.(U)

F/G 9/2

APR 77 W E BRANNING, D M WILLSON

F30602-76-C-0136

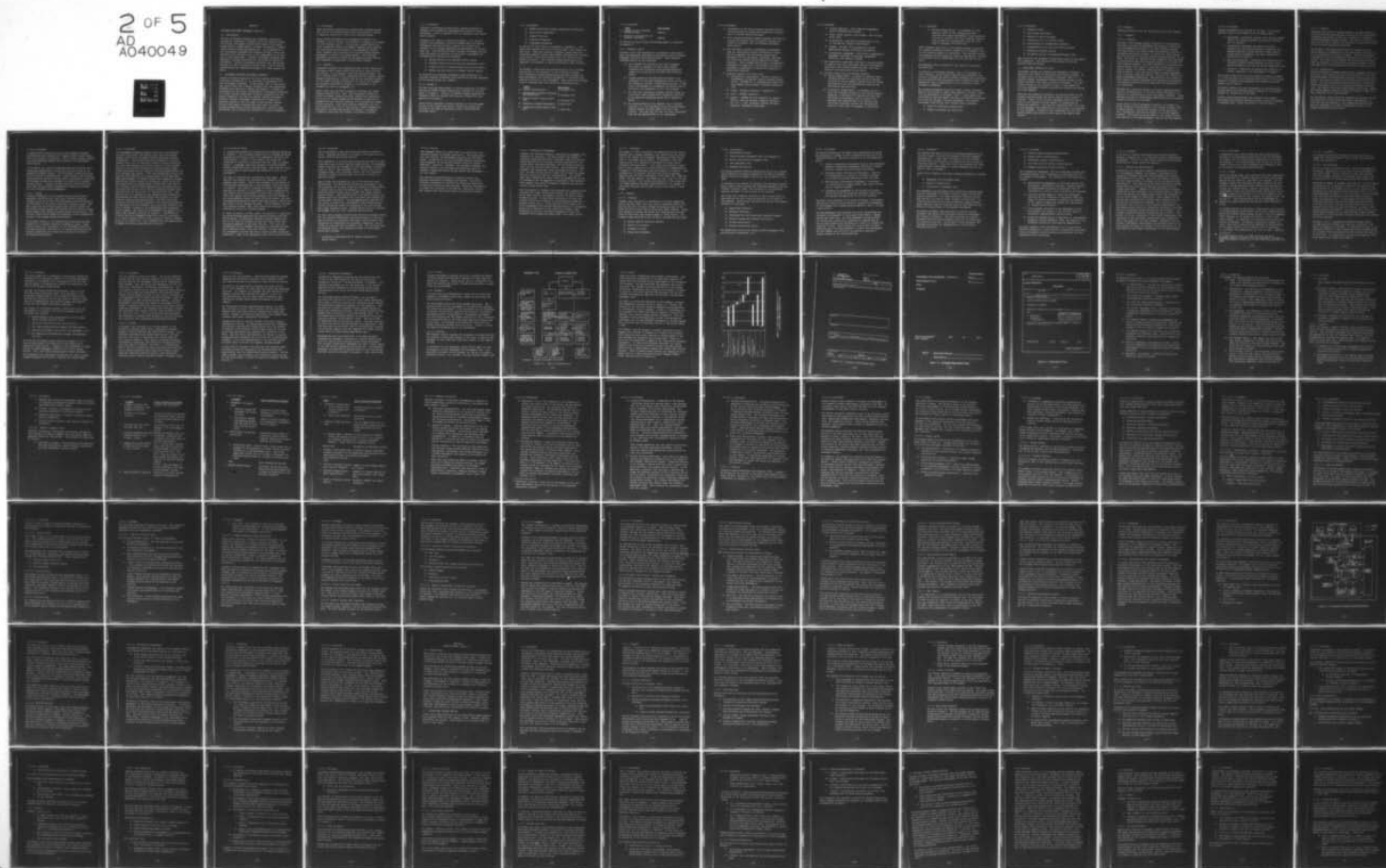
UNCLASSIFIED

PX-11932-F

RADC-TR-77-106

NL

2 OF 5
AD
A040049





SECTION 3

SOFTWARE DEVELOPMENT STANDARDS (TASK 2.2)

3.0 INTRODUCTION

Software Development Standards are standards, conventions, groundrules, guidelines, procedures and tools employed during the software development process which benefit the programs design quality, coding quality, software reliability, software viability and software maintainability. The Documented Software Development Standards were prepared by Sperry Univac and approved by the customer prior to the commencement of the process to which they apply. Software Development Standards also appeared as internal directions for use during the period of performance. Software Development Standards were specifically tailored to fulfill particular program needs, such as uniform identification and commentary information of coded program elements.

3.1 DOCUMENTED SOFTWARE DEVELOPMENT STANDARDS

During the schedule spanned by the four program developments there was a recognition of the importance of documented software development standards. In the earlier part of this period only ground rules were formally documented and some engineering standards were imposed by contract requirements. Program design guidelines and coding conventions were informally documented. As software systems became more centrally integrated from components supplied to the Navy by separate Program Development Agencies, software development standards were broadened in scope and adapted to meet the needs of each software system. Software development standards have been proven to ensure uniform program design, uniform documentation, uniform coding conventions, and to reduce the difficulty of system integration.

3.1 (continued)

These documented standards are valuable when followed through the life of the software because they are designed to produce a consistent definition to the entire software system.

Sperry Univac performed as the integration contractor for Programs 1 and 2, and was a subcontractor to support the integration for Programs 3 and 4. In each case, Sperry Univac supplied documented software development standards which improved the cost/schedule effectiveness to the integration process. These standards are continuing to be followed during program maintenance to assure the program quality. The following paragraphs describe in detail the documented software development standards as applicable to each studies Program.

3.1.1 Program 1

Documented standards were generated as part of the research and development model Common Program MOD XØ and X1 which provided the baseline to the Program 1 Operating System software. The Navy's NAVSHIPS 0967-011-0011 Specifications for Digital Computer Program Documentation, 1 October 1968 was the required standard for the Common Program MOD XØ and X1, as well as for Program 1. The documents delivered on Program 1 in accordance to the standard are listed in paragraph 1.8.

Sperry Univac also provided documented Guidelines for Common Program, Technical Note 301A. This guidelines document detailed the Common Program functional capabilities, guidelines for use of the Common Program for development of application programs and user reference material for interfacing the Common Program functions. From these baseline guidelines, the Software Guidelines for Users of the Program 1 Common Program were delivered to the Navy and provided as detailed information to agencies developing the Program 1 application programs.

3.1.1 (continued)

Therefore, the Navigation application program developed for Program 1 was designed and programmed with respect to these guidelines. Refer to Appendix B for an overview of this guidelines document.

Throughout the course of developing Program 1, the software analysts and programmers formed an informal media for passing information that was reviewed, updated and accepted as a standard for the design, generation and maintenance of the program code. The resulting informal Software Development Instructions were given as direction to all project programmers and included papers covering the following:

- a) Design and use of the resident control module.
- b) Standard format and use of software maintenance logs.
- c) Format and use of software debug plans and procedures.
- d) Format, use and control of software trouble reports.

As described in subsequent paragraphs, these guidelines become more formally documented, more definitive and more important as the systems become more complex.

3.1.2 Program 2

The Navy directed Sperry Univac to provide guidelines for implementing the NAVMAT Program Documentation Standard as it would apply to Program 2. Thereby, the NAVMAT Program Documentation Standard became the required Navy approved standard for documentation on Program 2.

Sperry Univac performed a detailed analysis of the development process of the Command and Control Operational Program and provided to the Navy an engineering report considering the following topics:

3.1.2 (continued)

- a) NAVMAT Operational Program Documentation Guidelines.
- b) Input/Output Formatters.
- c) Program Structure.
- d) Compiling Techniques.
- e) Programming Guidelines

An overview of each report is presented in Appendix C. This report was completed in June of 1972 and part 1 was updated one year later. Since the Program 2 Interface Design Specifications were completed before these NAVMAT Guidelines were written, the Interface Design Specifications were not affected by the NAVMAT Guidelines. (The Interface Design Specification was completed in accordance to Weapon Specification 8506, Revision 1.)

The Program 2 system test and evaluation group wrote informal instructions to aid generation of the Simulation Program. These Simulation Program working papers provided informal design requirements and interface information to coordinate and simplify directions to the programmer. These papers were written on an as required basis and covered the following topics:

<u>Topic</u>	<u>Date Issued</u>
a) General Specifications	18 February 1972
b) Supplementary Software Standards and Conventions	15 November 1972
c) Monitor Control Console Interface	20 November 1972
d) Simulation Program Maintenance	26 March 1973
e) Simulation Program Support Software	8 June 1973

3.1.2 (continued)

<u>Topic</u>	<u>Date Issued</u>
f) Command Console Interface Standardization	Unknown
g) Operator's Guideline for the Scenario Controller	Unknown

An overview of each of these six working papers is presented in Appendix D.

3.1.3 Programs 3 and 4

Programs 3 and 4 were developed in accordance to the requirements and guidelines presented in the supplied System Software Management Plan and System Specification. These documents contained details on the following:

- a) All programs were written in the CMS-2 languages and coded in accordance to the System Software Standards and Conventions using program generation capabilities specified by the CMS-2 User Reference Manual.
- b) Program generation was performed according to the techniques of "Top Down Programming" which provide for the coding and checkout of the program by levels. These levels corresponded to intermediate deliveries used to facilitate the integration of the total program.
- c) All real-time operational programs operated under the refurnished Common Program with interfaces specified by the Executive Operating System User's Reference Manual.
- d) The program adhered to the budgeted use of computer resources, i.e. CPU allocation in milliseconds/second, computer memory, and disk memory allocation and accesses. This was done for both the fully operational mode and the degraded mode of all subprograms.

3.1.3 (continued)

- e) Conformance to the System Software Standards and Conventions during the program development was a requirement of Software Quality Assurance.
- f) Each developer performed computer program qualification testing and verified that the computer program performed in accordance with the Program Performance Specification.
- g) Each developer provided a Configuration Management Plan for maintaining internal configuration control and for complying with the System Configuration Management Plan in proposing, evaluating, approving and effecting computer program Engineering Change Proposals. After program qualification, configuration management was applied for the program life cycle in accordance to guidelines to ensure the integrity of established baselines.
- h) Standards imposed on software were:
 - 1) Documentation requirements - WS-8506, Revision 1, Requirements for Digital Computer Program Documentation; TADSTAND No. 3, Standard Requirements for Inter-Digital Data Processor Interface Documentation.
 - 2) Format - WS-8506, Revision 1; TADSTAND No. 3
 - 3) Style - WS-8506, Revision 1
 - 4) Symbols - NAWSEPS OP 1700 Standard Fire Control Symbols; American National Standards Institute, Inc. X3 5 - 1970, Flowcharts Symbols and Their Usage for Information Processing

3.1.3 (continued)

- 5) Display symbology - Joint Computer Programming Centers Standardization Manual
- 6) Words, terms, phrases, and acronyms - ADP Glossary, NAVSO P3097
- 7) Abbreviations - MIL-STD-12 Abbreviations for Use on Drawings and Technical Type Manuals
- 8) Changes - MIL-STD -480 Configuration Control Engineering Changes, Deviations, and Waivers
- 9) Configuration management - MIL-STD-483 Configuration Management Practices for Systems, Equipments, Munitions, and Computer Programs
- i) Each application subprogram consisted of program modules to comprise the fully operational mode and the degraded operational mode system program such that vital ship functions were maintained in both modes.
- j) Software design reviews performed included:
 - 1) System Design Review - to ensure a technical understanding between the contractor and the procuring activity on system segments identified in the System Specification and the configuration items in the performance specifications.
 - 2) Preliminary Design Review - to evaluate the progress and technical adequacy of the selected design approach, to determine its compatibility with the performance requirements, and to establish the existence and compatibility of the physical and functional interfaces between the configuration items and other items of equipment or facilities.

3.1.3 (continued)

- 3) Critical Design Review - to determine that the detailed design of each configuration item satisfies the design requirements in the specification and to establish the exact interface relationships between the configuration items and other items of equipment and facilities.

These standards were vital to the program success, especially for complex integrated software systems such as Programs 3 and 4 which involve a number of contractors. The standards assured the developed system program was real-time operational, operated as specified, and was maintainable and low cost for its life cycle.

See Appendix E for an overview of the content of the System Software Plan.

In addition to developing Programs 3 and 4 in accordance to these standards, Sperry Univac supported the integration contractor in forming the System Software Standards and Conventions, and the Executive Operating System User's Reference Manual (see Appendices F and G, respectively, for an overview of these documented standards).

The program development groups also wrote informal internally applied instructions to improve and simplify the design, generation and maintenance of Programs 3 and 4. These papers were identified by name and number, and were collected and maintained as a reference document for each programmer. The Software Development Instructions (SDI) covered topics including:

- a) Rules for defining a design object.
- b) Example procedures for using RIPS.

3.1.3 (continued)

- c) Coding of stubs.
- d) Tape naming conventions.
- e) Meaning of control messages.
- f) Disposition of to be defined items.
- g) Instructions for documentation updating.
- h) Documentation of St. Paul site system program.
- i) Handling of program listing.
- j) Procedures for use of program maintenance logs.

This collection was expanded by submitting a paper to the groups' SDI coordinator. Upon review by key personnel an SDI was finalized and distributed.

3.2 DEVELOPMENT METHODS AND TOOLS

Development methods are technical and managerial procedures employed at the stages of developing each computer program. Development tools are any available or developed items (i.e. a computer program, programmed hardware/software system, or documented procedure) that benefit software development. For each studies Program at each identified stage of development the following paragraphs identify, describe and evaluate the methods and tools employed.

Technical and managerial procedures are usually dictated by the contract procuring agent and also by Sperry Univac contract policies. Sperry Univac internal management practices are defined in the Standard Practices Manual for Engineering and Production Operations. The task and responsibilities of all key project personnel are established for each project in accordance to this standard. Included in this requirement are the documented project plan, monthly status reports and weekly flash reports.

3.2.1 Program 1

Detailed information was not made available for this program.

3.2.2 Program 2

3.2.2.1 Planning

The Navy's C&CS program manager developed the C&CS Management Plan which outlined the management and technical controls for the development and delivery of the C&CS computer programs. This plan defined the overall tasks and milestones required to accomplish the project efforts, to ensure the timely development and delivery of the C&CS computer programs required for the ship. This plan was reflected in the contracted work performed by Sperry Univac and other contracted agencies to provide the C&CS computer programs. The Navy monitored all contracted work status monthly with respect to this plan and maintained this plan to reflect the current status and plan to complete the project efforts. This plan was the Navy's overall management tool to provide status and control. The Navy C&CS computer programs procurement organization then provided a more detailed C&CS Computer Program Development Plan defining all milestones, the milestones schedule and how they would be accomplished by the contractors.

The Navy also conducted periodic co-location meetings participated by representatives of the C&CS Program development contractors for coordination of their development schedules. These meetings provided for maintaining communication and coordination between contractors and the Navy.

Sperry Univac internal management practices for Program 2 required formation and maintenance of the Program 2 Project Plan as Sperry Univac's controlling document for the development, test, integration, delivery and support of the C&CS computer programs. This project plan was based on the overall development approach of the C&CS Management Plan and the C&CS Computer

3.2.2.1 (continued)

Program development Plan provided by the Navy. This project plan was bound in four volumes in the following topics:

- 1) Development - described, through use of defined planned events, the detailed approach for efficient development of Program 2.
- 2) Configuration Management and Quality Assurance - specified the plans and procedures to establish firm program baselines, provide positive change control, insure quality workmanship, and enact comprehensive validation of Program 2.
- 3) Test Site Operations - delineated the procedures to be followed by Sperry Univac personnel at the St. Paul program generation site and the system integration site.
- 4) Financial - Summarized the manpower and financial resources committed to the development of Program 2.

This project plan served well the planning to complete Program 2. It was the means for introducing this project by the Project Engineer to Sperry Univac management and thereafter provided monthly status reports.

The Project Engineer was also responsible for monitoring engineering status and reporting status bi-weekly to the Contract Manager for transmission to the Navy.

Software engineering personnel also completed the Program 2 Maintenance Investigation Engineering Report during the Program 2 planning stage. (Refer to section 9.2 of this report.)

3.2.2.2 Analysis

Throughout the period of program analysis, the Navy conducted Design Review Task Group meetings periodically for participation by program development contractors to define all interfaces between programmed computer systems. The results of these meetings were the early complete and compatible interface definitions in the Interface Design Specifications.

The Program 2 development group wrote the Interface Design Specifications in preparation for the development and validation, for the navigation by satellite program. They also generated a navigation simulation program for execution on the Univac 1108 system. This tool proved excellent for developing and validating the equations of the satellite navigation program, and was used again later for developing the satellite simulation model.

3.2.2.3 Specification

During the period of program specification, management visibility and control were formally provided through reviews by the Navy procuring agency with development group participation and internally by the project technical coordination group within Sperry Univac. The Interim Program Reviews conducted by the Navy provided preliminary review of the program performance specification. The reports from these reviews defined additions, changes and further analysis performed during this period to meet the Navy acceptance of the specification.

The Preliminary Design Review was conducted by the Navy, following completion of the program performance specification, to establish the Navy approval of the program specification and mark the beginning of the Navy configuration control of the program performance specification.

3.2.2.3 (continued)

In Sperry Univac internal reviews of the program performance specification were conducted by a separate project group responsible for technical coordination. These reviews assured completeness and consistency between all sections of the performance specification.

In retrospect, the Interim Program Reviews were excellent by establishing communication, control and visibility early in the program development process. The Preliminary Design Review also proved excellent by establishing a firm milestone, and providing a specification baseline that was easily configuration controlled. The internal reviews were necessary for delivery of a quality assured product. Overall, the most effective specification reviews were those providing communication between the customer and the development group programmers.

3.2.2.4 Design

During the generation of the program design specifications, management visibility and control were provided externally through reviews by the Navy with development group participation and internally by the project technical coordinator group. The Navy conducted the Interim Program Design Reviews against the preliminary program design specification and the Critical Design Review against the completed program design specifications. The Critical Design Review marked the beginning of Navy configuration control of the program design specification.

The internal reviews of the program design specification were conducted by a separate technical coordination group. These reviews assured the requirements of the performance specification were adequately satisfied in the design specification, and assured completeness and consistency between all sections of the design specification.

3.2.2.4 (continued)

The programmers designing Program 2 used two tools developed for use on the Univac 1108 system; Trace Inputs Processing Outputs (TIPO) and an automated flowchart generator program (FLOWCHARTS). TIPO accepted data files generated by the programmers based upon the specified Program 2 design identifying all input data messages and external stimulus to each program module, all procedures referenced by the subsequent program module processing, and all possible resulting program module outputs (messages and signals). TIPO then automatically traced the possible paths taken by the program design for any identified input data message or external stimulus that triggered a program function and listed in order all procedures within various program modules and all program module outputs that could result. This listing would show upon examination any discontinuities in processing that should result from any designated program input and would give the big picture of functional data flow. This analysis led to a redesign of some message traffic between program modules and logic within modules and resulted in a more efficient logic and processing design. Had TIPO lists been accepted as functional flow diagrams in the Design Specification and maintained with the document, they would have served for the life of Program 2 adding to the understanding and maintenance of the program. The second tool, FLOWCHARTS, was applied to all newly coded programs developed on Program 2. FLOWCHARTS constructed detailed subprogram design flow charts in accordance to the ASCII standard directly from specially formatted comments given in the source code. With FLOWCHARTS the subprogram design documents were easily generated and maintained.

3.2.2.5 Code and Debug

A conventional process was followed during coding and debugging of Program 2. The successful production of a debugged operating program was attributed in part to the management procedures and tools applied. A result of the analysis and design was the allocation of functions to software tasks and tasks to program modules. Programmer teams were assigned to produce program modules having functionally related areas. Each team was led by a section head of senior status and consisted of five or six programmers.

Two forms of reporting were provided by these teams. Status logs and core reports were maintained and updated by the programmers upon request. Status logs contained the percentage of code produced and percentage of code debugged on each procedure of a module. Core reports contained the number of memory words required for allocation to the segments of each program module. Core reports were re-established for each module recompile. At first, core reports were provided manually, but after Program 2 was being generated using the Univac 1108 RIPS system, they were automatically generated by predictions made directly from the program object code as changes were made. Since Program 2 operated from computer resident code, these reports were essential to reducing the risk of an oversize program.

As shown in Section 1 of this report, the majority of compilation and system program generation was completed using an off-line CMS-2Y system. An individual dedicated computer operator ran most of the jobs for Program 2 during this time, rather than allowing programmers to run their own jobs. This, plus the standardized use of SYSMAKER (see appendix D) provided an efficient operation using a tape-based system. For many reasons this Government furnished tape-based program support system was inefficient. Many of the peripherals were old, slow and worn-out,

3.2.2.5 (continued)

and the handling of tapes was tedious and prone to operator error. Therefore maintenance of Program 2 was moved on to the Univac 1108/RIPS system when possible.

After generation of a system tape containing the executive operating system and a newly compiled program module, the programmer loaded, executed, and debugged the code with hands-on equipment access in the St. Paul equipment suite. Before this on-line debugging began, each programmer assured that the program was logically complete and relatively error free by careful inspection process.

When on-line, the programmer had use of the operating system Debug Module. The Debug Module allowed the programmer to control execution of the tasks within an identified program module, to extract data at selected breakpoints, to measure program execution point-to-point, and to inspect and change computer memory contents. Upon satisfaction of the debug criteria, the program module was integrated onto the system tape prepared for testing under simulation. The programming group supported the test group to resolve all discrepancies found during testing. Initial testing was really the beginning of the next development stage, program testing. Program modules were not considered debugged until all tests completed satisfactorily.

The test system program provided some additional debugging tools. The test system automatically controlled execution of the program functions in a simulated environment, made automatic data extractions at established and selectable breakpoints, and recorded the extracted data on magnetic tape. The resulting tape was analyzed using a data reduction program to obtain appropriate data listings.

* - Information concerning RIPS is Company Proprietary to Sperry Univac.

3.2.2.6 Testing

Each debugged module was turned over to the system test and evaluation group for program verification. Two organizations were now involved, the development group and the evaluation group. In addition to the tools already available during program debugging, an interactive display controlled scenario generator - extractor - recorder - and automatic playback, the Scripted Scenario, was developed and extensively used for sequence, timing and control during testing (see paragraph 4.... for more detail).

The evaluation group maintained a detailed record of all problems encountered in hardware and software. The record contained information to identify the conditions under which the problem occurred and when possible, sufficient information to repeat the occurrence of the problem. Evaluation testing was complete when this record showed no unresolved problems.

3.2.2.7 Configuration Management

Program 2 was provided interrupt configuration management from beginning to completion. The Navy accepted the Program 2 Interface Design Specification, Performance Specification and Design Specification following criteria established by the design review task group meetings, the preliminary design review, and the critical design review. Upon acceptance, these documents were placed under Navy configuration control. For this, the Navy established the Change Control Board including Navy and contracted technical representation. Any software under Navy configuration control could be changed only by and in accordance with the approval of this board. The records of these changes were called Preliminary Design Review Logs and Critical Design Review Logs.

Once the modules of Program 2 were accepted for integration and delivered for formal verification testing, the modules were placed under internal configuration control by Sperry Univacs integration group. During formal verification testing, any software problem was submitted to the maintenance agency as a program trouble report. These logs and trouble reports provided recommended solutions and when approved by the integration group would become part of the controlled software. Upon acceptance by the Navy following integration testing, the integrated C&CS program was placed under strict Navy control.

3.2.2.7 (continued)

Any change to a Navy controlled item needed or desired, but not apparent until program completion, was submitted by the originator of the request as Preliminary Engineering Change Proposal (ECP) to the Change Control Board. If desirable, ECPs were usually given to a responsible contractor for study and resubmission with more detail. This study would provide a statement of need, suggested or alternate methods for implementation, statement of risk assessment and cost/schedule to complete. Upon resubmission, if approved, the work task would be assigned, completed and the result documented in the formal ECP. When the formal ECP was approved, the change was then a part of the controlled software. The records and operations of the Change Control Board provided excellent identification of all changes to controlled baselines and provided an up-to-date status of each change.

3.2.3 Program 3

3.2.3.1 Planning

Program 3 was received as a subcontract to provide Command and Control Systems (C&CS) real-time tactical operating system software and was over two years through the planning stage by the prime contractor before Sperry Univac came under contract. Therefore extensive planning and system level descriptions were provided as directions when planning began in Sperry Univac. The documents made available by the customers included the following:

- a) System Proposed Technical Approach.
- b) Request for Proposal.
- c) Statement of Work.
- d) Design Data Documents.

3.2.3.1 (continued)

- e) System Specification.
- f) System Software Management Plan (see appendix E).
- g) System Configuration Management Plan.
- h) Data Management Plan.
- i) Degraded Mode Report.

The System Proposed Technical Approach was one of the original system definition documents prepared by the Navy procurement agency defining the overall requirements and operation of the C&CS.

In response to the Request for Proposal and referenced requirements, Sperry Univac provided the Technical/Management and Cost Program 3 Proposal, which later resulted in the negotiated supplied Statement of Work as part of the contract.

The design data documents consisted of many volumes which became available as each volume was approved by the Navy for distribution. These system definition and overall requirements included the following:

- a) Operational Narrative.
- b) Hardware Interfaces.
- c) Functional Flow and Operational Sequence Diagrams.
- d) Software Design and Interface Description.
- e) System Design Report.
- f) Function Operational Design.

The System Specification and System Software Management Plan are discussed in paragraph 3.1.3.

3.2.3.1 (continued)

The System Configuration Management Plan established the policy and procedures to manage the C&CS configuration and that of the evaluation facility. This plan consisted of the following four parts:

- 1) General Configuration Management Policies and Procedures - standard configuration management functions of identification, change control and status accounting.
- 2) Interface Control Policies and Procedures - relative to documentation and control of interfaces external to the C&CS and between subsystems within the C&CS.
- 3) C&CS Software Configuration Management - procedures for identification, change control, and status accounting for computer programs.
- 4) Configuration Management of the Evaluation Facility - control policies for shipbound equipment being tested, for resident equipment and for the test facility.

Only part 1 was available at the onset of Program 3 development, and over the course of constructing the configuration management of both hardware and software during integration, parts 2 through 4 were written.

The Data Management Plan provided the policy, methodology and authority for management of the program deliverables, communications and other data. The data management system provided for control and processing of incoming and outgoing data as well as data storage with procedures for control of storage and retrieval of data. Sperry Univac also established and maintained as part of program management a similar data management. The tool MAPPER (see section 9) was applied to maintain a current catalog to all stored materials.

3.2.3.1 (continued)

The Degraded Mode Report provided a documented analysis of the continuous support from the C&CS program in multiple hardware configurations. A major part of Program 3 was the software which provided initialization, reconfiguration and recovery of real-time operation. Since this report was reissued four times during Program 3 development, it caused redesign from the top down of part of Program 3.

Sperry Univac supplied three major planning documents for Program 3:

- 1) Management and Performance Plan.
- 2) Quality Assurance Plan.
- 3) Configuration Management Plan.

The Management and Performance Plan provided the detailed planning for development of Program 3 and for support of the C&CS program integration. This plan satisfied a contract requirement and was the internal formal Program 3 project plan. Included in this plan was the work description, administration plan, financial plan and technical plan for efficient and cost effective accomplishment of the program development.

The Quality Assurance Plan described the software quality assurance that Sperry Univac provided during the development and testing of Program 3. This plan described the quality assurance objectives and procedures instituted to assure that the objectives were met. The software quality assurance disciplines was obtained by implementation of the following described controls:

3.2.3.1 (continued)

- a) Software quality program establishment.
- b) Design control identification.
- c) Development control identification.
- d) Test/Certification control identification.
- e) Program records and change controls.

The Configuration Management (CM) Plan described the plans and procedures applied to all configuration items identified for the Program 3 development. The objectives of this plan were as follows:

- a) Assist technical management in achieving required item performance, operational efficiency, logistics support, and readiness by providing the necessary level of configuration, control, status accounting, and auditing.
- b) Allow the maximum degree of design and development latitude, and yet introduce, at the proper time, the degree of control necessary to maintain the integrity of the computer program and satisfy the requirements of baseline specifications.
- c) Attain maximum efficiency in the management of software changes with respect to the cost and timing of evaluation, implementation, and recording.
- d) Attain the optimum degree of uniformity in configuration management policy, procedures, forms, and reports of all interfaces between all participating organizations.

The major segments of the plan addressed the CM organization, Software Configuration Control Boards, configuration identification, configuration control policies and procedures, and configuration status accounting.

3.2.3.1 (continued)

As dictated by Request for Proposal and accepted Technical Proposal, Program 3 was to be developed using top down program development. Section 2 of this report describes the processes established by Sperry Univac for Program 3.

3.2.3.2 Analysis And Specification

Working from the specified technical requirements provided by the planning documents, the Program 3 development group completed an analysis and wrote the preliminary performance specification in six weeks. While this performance specification was being reviewed, work began on the preliminary design specification. Four weeks later the first C&CS Program System Design Review was held by the Navy and participated by all Program contractors. Comments on the preliminary performance specification were submitted to the customer before the meeting. During the meeting the comments were reviewed and answered or left as action items for resolutions. These comments and responses were collected into a review report prepared for this meeting and distributed to attendees. As a result of this meeting and some revisions to the technical requirements, the performance specification was revised and delivered at the same time the preliminary design specification was delivered. The Program 3 generated from this design was called level 1. This process of review, alteration of some overall requirements, and reissuance of the performance specification was repeated for level 2 and again for the final program. Therefore, even the Program 3 performance specification was developed from the top down through revisions, and became a better description of the functional and performance requirements of the final program.

3.2.3.2 (continued)

A significant factor that allowed for rapid generation and revision of the performance specification was the modularity of Program 3 and the working group organization as described for Program 2. The program modules were defined by the performance specification and programmers were assigned to each at the onset of the program design process.

3.2.3.3 Design

During the design stage, each specified functional requirement was interpreted as a function supported by one or more program modules and was provided a functional flow diagram. All performance capabilities specified for each program module were provided a design detailed by data designs and executable procedures with defined inputs, outputs, processing and special requirements. Since the resulting design specification was to describe generation of Program 3 in a top down manner, a special documenting construct was invented, the design object (see section 2). Each program module was then diagrammed as a network of design objects and each design object described in detail.

In a manner similar to the review, review report, and revision of the performance specification, the Program 3 design specification was provided formal design reviews by its user community before approval of the design for the level 1, level 2 and final program. It, therefore, was provided review reports and revised accordingly. The design specification prepared for level 1 and level 2 included as an appendix an identification of the design object is to be coded in full, or partially, i.e. stubbed*, for completion of the level of code. Therefore,

*A stubbed design object is coded to utilize its final estimated computer resources (CPU time and memory), accepts input and produces outputs, but is not coded with any specified logic.

3.2.3.3 (continued)

the design specification described in detail the coded, debugged and verified program for each leveled delivery ahead of any code generation.

3.2.3.4 Code And Debug

While the design specification was in review before each level of Program 3 generation, the top down coding and debugging of each module began. Before turn-over of each coded program level the changes described by the design review report were incorporated into the design specification and program code.

Coding and debugging of Program 3 proceeded in a top down manner. The intention was to produce a skeleton of the specified completed design for the program in code, so that program coding, debugging, test and delivery would take place early in the program development schedule. In fact, the level 1 code of Program 3 was delivered and installed in six facilities six months after work authorization. In this manner the C&CS Program subsystem developers received the operating system software in leveled deliveries and each delivered level provided the necessary and sufficient executive software to support their leveled program developments. This gave an early start to the C&CS Program integration such that any flaw in major program design would be identified and corrected early while the program was being developed.

Program 3, level 1 contained all specified design objects. Although many were stubbed, all specified interfaces to the executive program were provided. At completion of level 2, all interfaces external to each program module were provided using the intended format and appropriated the sequence and condition of occurrence. At completion of the partial program all functions specified in design were complete.

3.2.3.4 (continued)

The final program was a refinement of the partial program and complete for all functions required for the subsystem program developers to accomplish their development and for the integration agency to integrate and evaluate the operational test facility development program model.

Although the Program contracted development came to an end with this final development model, as an extension to the integration support tasks to this contractor, Sperry Univac developed and delivered the sea test model of Program 3 based upon the recommendations and required changes made apparent during the C&CS Program integration evaluation.

The success for developing Program 3 rapidly and in the top down manner is a tribute to the project personnel and the following tools applied:

- a) Modular design using Design Objects, and the Design Object Drawing Outputter (DODO).
- b) The Univac 1180/RIPS system (see section 9).
- c) The Common Program Utility Package and Debug Module.
- d) Software Development Instructions (see paragraph 3.1.3).
- e) The Simulation and Evaluation Program (see paragraph 3.2.3.5).

Not only did the design objects direct the coding to a low level design element, the definition of each module was a network of design objects. Program 3 as an integration of these modules provided management visibility and control during product. Also, each modules design object network provided a true diagram of the coded module that proved very useful during program debugging and testing to identify each coded element

3.2.3.4 (continued)

and its interfaces within the module. The Univac 1108/RIPS system used for text editing, compiler, librarian and loader greatly improved the programmer effectively by supplying demand terminal, time sharing and data management for the code production, maintenance, and machine code generation. The tools proven valuable for all program using the Common Program, the Utility Package and Debug Module, were enhanced during Program 3's development and used at each level in accordance to the programmer's written debug plan and procedure to show proper operation of each specified modular function. Software Development Instructions were used as an instrument for exchange of generally applicable ideas and to coordinate efforts among the programmers. The Simulation and Evaluation Program was integrated with Program 3 and the Common Program in the St. Paul site suite. With this system each leveled program was subjected to simulated C&CS Program use and thus verified the correctness and completeness before program delivery.

3.2.3.5 Testing

As Program 3 was developed in a top down manner, it was tested by a separate Sperry Univac organization assigned Program 3 quality assurance and qualification testing. Those functions specified in the design specification to be completed at each level of generation were tested using the test program composed of test control, subsystem simulation, intercomputer simulation, data extraction recording and off-line data reduction modules. For level 1, a recording was made of all task program calls upon the executive as the test control module operating from a card input scenario stimulated the subsystem simulation modules, thus driving the operating system software including Program 3 through a stimu-

3.2.3.5 (continued)

lation of all its interfaces. Then the data reduction program formatted and printed the contents of this interface trace. This printer listing was provided as a part of the test record.

For level 2, the same system of modules were used, and the level 1 test repeated, but with many new features and additional functions. For example, all messages sent and received by the Program 3 modules were recorded and compared to their design specification, and the inter-computer interfaces were functionally checked using simulation programs in another computer.

For the partial and final programs qualification, not only were the tests of level 1 and 2 repeated, but all interfaces together with their data accuracy, sequence and timing were tested. Thereby, all functions specified by the program specifications were shown to qualify as deliverable.

Through each level of testing, Program 3 was under internal configuration control. No design modifications nor recompiles were allowed, and corrective modifications were only allowed in response to the software trouble reports given by the test group using maintenance logs approved for correction of the error. When all tests were satisfied, the modules in need of recompiling were recompiled and the entire test satisfactorily repeated. It was the responsibility of the test and quality assurance group to deliver the Program 3 program package.

At each level of testing a test plan and test procedure document was provided and approved by the customer before the qualification test was completed and results reported. For the final program, these documents were formalized, edited and delivered in accordance to NAVORD Documentation Standard, WS-8506, Revision 1.

3.2.3.6 Configuration Management

Configuration Management planning was well defined and controlled on the C&CS Program by the Navy and the system integration contractor. During development of Program 3, configuration management was the responsibility of the project engineer and was delegated to the development group during code generation and to the test group during qualification testing.

The development group installed Program 3 at six remote facilities for each of two or three of the leveled deliveries. These remote facilities provided the operating system software for application program developments by other agencies. Each facility was provided a program listing, program operator's manual and installation notebook. The quality assurance group managed these configuration controlled program items. The development group provided support for resolution of all user submitted Software Trouble Reports (STRs). The log of STRs was maintained for the program as a whole. STRs were resolved in source code before the next program level generation. STRs from each facility were made a part of the facilities installation notebook, together with the temporary fix (patch) for any coding errors by using program maintenance logs.

Following delivery and installation of the final program, configuration management continued at a reduced manning level using the controls and procedures proven during development. This extended configuration management has been a direct support to the C&CS Program integration and evaluation. Configuration management was supported by Sperry Univac for both the development model and sea test model of Program 3. Tools used during

3.2.3.6 (cont.)

program development including the Univac 1108/RIPS and Simulation/Test system were applied to maintain the high technical quality of Program 3. Additionally, project information, documents and records were stored, cataloged and provided a current status by using MAPPER.

3.2.4 Program 4

To produce the qualified Program 4, Sperry Univac defined and applied a development process and a means for monitoring and controlling the process.

The development process had three primary activities. The principal activity was development of the application program. The second was development of the simulation and test program as a parallel activity which supported the checkout, integration, and qualification testing phases of the program development. The third activity was the support function for test, evaluation, and integration both at the land-based evaluation facility and on the ship. The exchange of documentation constituted the formal interaction between these activities. Figure 3-1 shows the chain of documents used in the development.

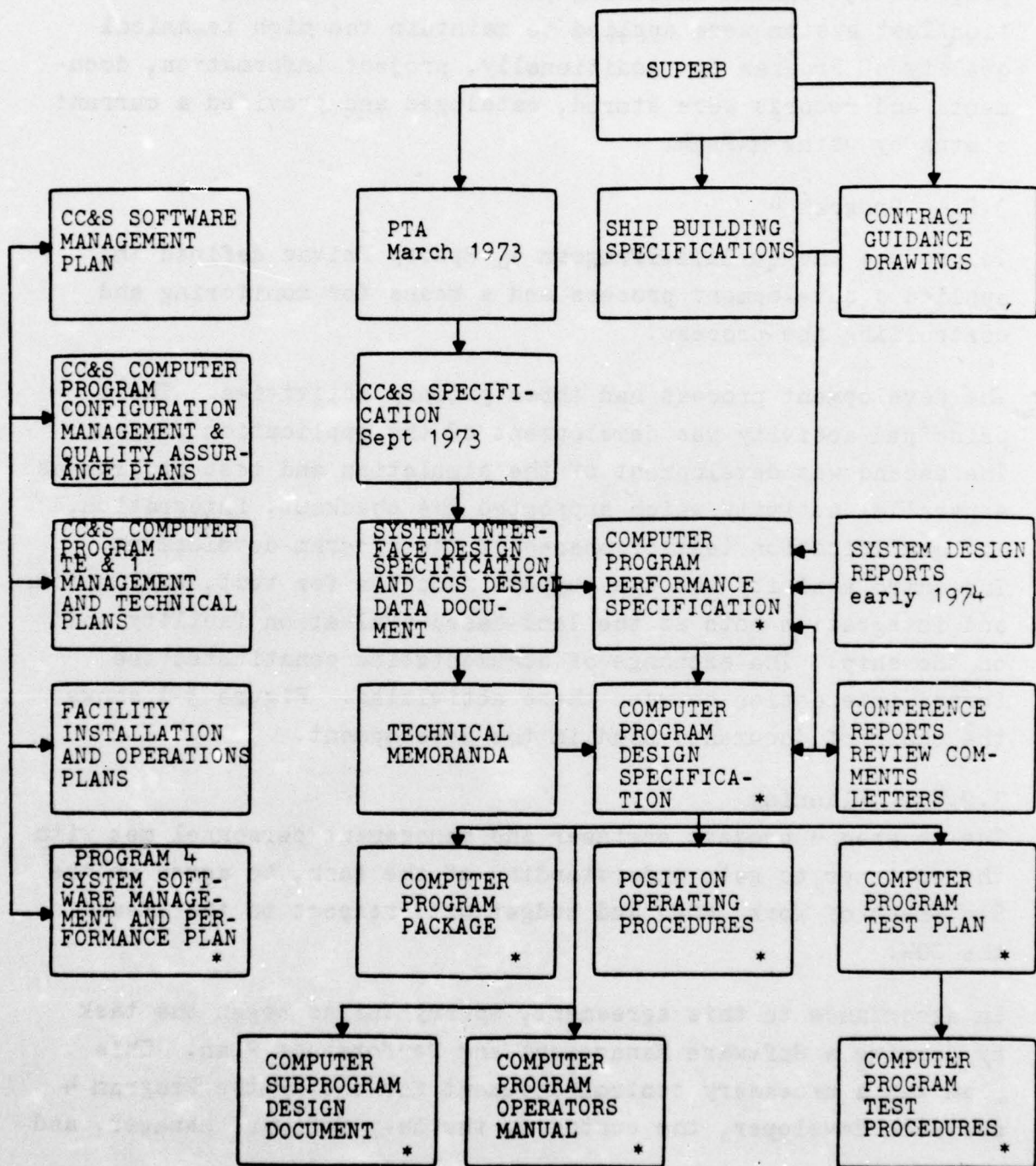
3.2.4.1 Planning

The Program 4 project engineer and management personnel met with the customer to gain understanding of the task, to agree on the Statement of Work (SOW) and budget with respect to the items in the SOW.

In accordance to this agreement, Sperry Univac began the task by forming a Software Management and Performance Plan. This plan was a necessary control document for use by the Program 4 software developer, the customer, the Navy Software Manager, and

MANAGEMENT PLAN

TECHNICAL DOCUMENTATION



* Program 4 Software Development Deliverable

Figure 3-1. Chain of Documentation

3.2.4.1 (cont.)

other activities interfacing with Program 4 development. This plan presented a description of the development approach to ensure low risk and fully visible development of the integrated C&CS Program. The plan described the management and control procedures that ensured successful completion of the design, production, integration, and testing of the Program 4 software. In addition, this plan supported the milestones for the C&CS software development as described in the Navy's C&CS Software Management Plan.

The major activities and schedules for the Program 4 software and integration tasks are shown in figure 3-2. Further subdivision of the task into subtask statements was accomplished by the Program 4 project engineer. Figure 3-3 illustrates the work package form used by the project engineer to distribute, identify, and control program subtask assignments. This form contained a brief work description, an end item description, references, and cost and schedule information to provide management visibility and to facilitate control of the total Program 4 task.

The Program 4 project engineer, customer, and Navy's account manager were informed of progress on bi-weekly status reports which identified milestones achieved in accomplishment of the work packages and documented any problem areas. Figure 3-4 shows the form completed bi-weekly by the responsible engineers/programmers. In cases where problems developed, the project engineer would institute corrective action to assure, problem resolution in a timely manner, meeting the overall C&CS problem schedule. These bi-weekly status reports were summarized in a monthly report to Ship Acquisition Project Manager (SHAPM) using the form shown in figure 3-5.

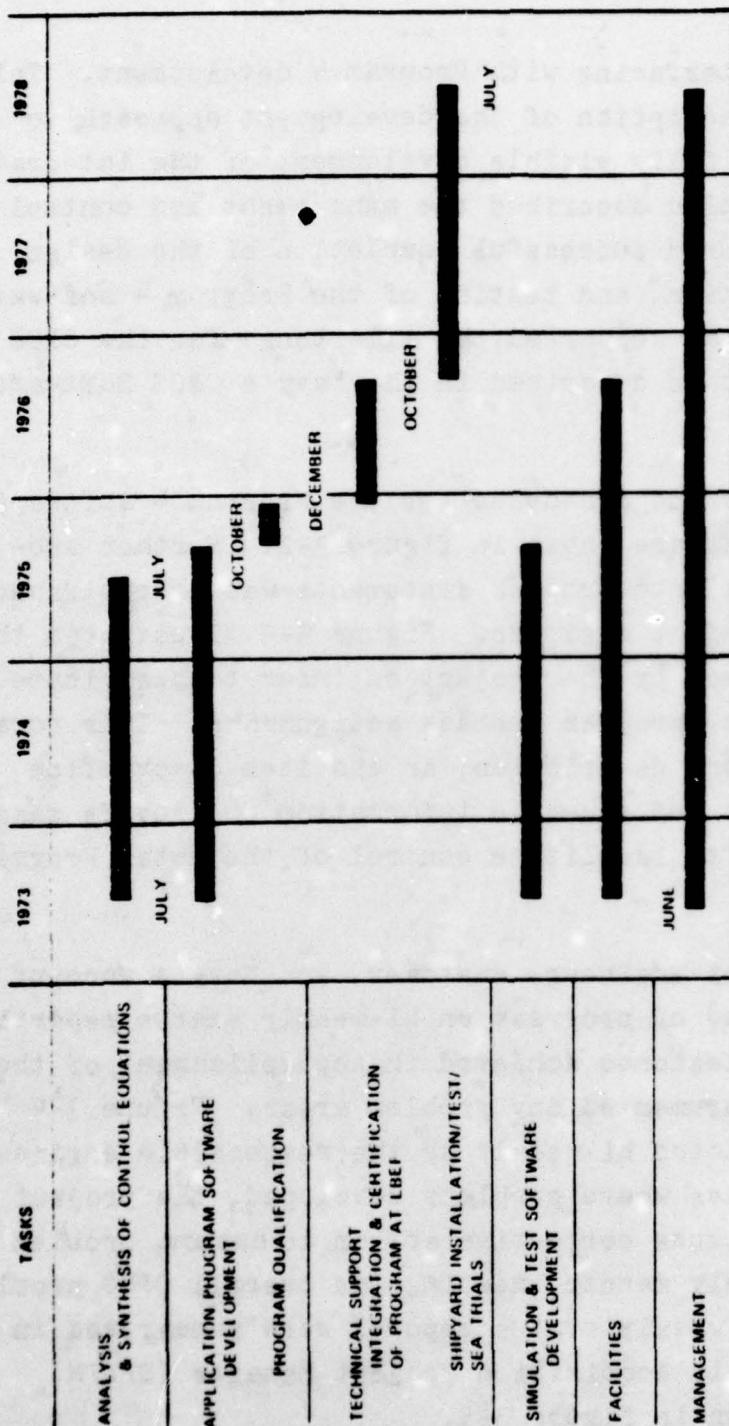


Figure 3-2. Major Tasks and Schedule for Program 4
Software and System Integration Task

<div style="border: 1px solid black; padding: 2px; display: inline-block;"> Program 4 Software and Integration Task </div>	DATE _____	
	PROGRAM TASK ASSIGNMENT _____	
	TASK _____	SHOP ORDER _____
RESPONSIBLE ENGINEER (PERFORMING DEPT) _____		DEPT _____ EXT _____

WORK DESCRIPTION

END ITEMS

REFERENCES

COST ESTIMATE	MANHOURS _____	COMPUTER HOURS _____	MATERIAL S _____
---------------	----------------	----------------------	------------------

SCHEDULE

AUTHORIZATION			
TASK ENGINEER	NAME _____	DATE _____	DEPARTMENT DOING WORK
	NAME _____	DATE _____	

Figure 3-3. Program 4 Work Package Form

BI-WEEKLY STATUS REPORT (PROGRAM 4)

Period of Report

From: _____

Task (Sequential No.): _____

To: _____

Title: _____

Progress: _____

Are you on schedule?
(If no, explain)

Yes

No

N/A

Signed: **Responsible Engineer** _____

Task Engineer _____

Figure 3-4. Bi-Weekly Status Report Form

TITLE <div style="text-align: center; border-top: 1px solid black; border-bottom: 1px solid black; margin: 5px 0;">REPORT. STATUS</div>	CONTROL NUMBER: DID <u>UDI-A-23000</u> DIR _____
---	---

SECURITY CLASSIFICATION _____

STATUS REPORT

Reporting Period _____ Year Month _____ Task WBS No. _____

Task Title _____

Name of Contractor/Government Agency _____
 Responsible for Execution of the Work _____

1. MILESTONES COMPLETED SINCE LAST REPORT: _____
2. MILESTONES NOT COMPLETED: _____
3. PERCENTAGE OF WORK (Current Fiscal Year) Accomplished To Date: _____ %
4. PRICE

a. Dollars Authorized	CURRENT FY	PREVIOUS FY
b. Cumulative Price		
c. Estimated Price to Complete Funded Task		
5. NARRATIVE OF TASK EFFORT DURING THIS REPORTING PERIOD: (If additional space is required, use the reverse side of the report)

6. _____
 Authorized Signature Code, Title Telephone No. Date

SECURITY CLASSIFICATION _____

Figure 3-5. Status Report Form

3.2.4.1 (continued)

As represented in figure 3-1, the Navy provided the following documents to support and direct Program 4 development:

- a) CCS Specification - Provided the system-level requirements which must be satisfied by the individual subsystems of the CCS and by any interface points between the CCS and other ship systems.
- b) CCS Design Data Document - Provided common design, functional, and resource allocation.
- c) CCS Interface Design Specification - Identified the CCS system and subsystem interfaces.
- d) CCS Software Management Plan - Provided the approach to planning, management, control, generation, testing, integration, and acceptance of the CCS computer programs.
- e) CCS Computer Program Configuration Management and Quality Assurance Plans and Procedures - Detailed the procedures and discipline necessary for controlling changes to the CCS computer programs and documents.
- f) CCS Computer Program Test Evaluation and Integration Management and Technical Plans - Identified the test sequence, schedules, and procedures for integration activities.
- g) Facility Installation and Operations Plans - Identified the preparation, installation, and operational requirements for all CCS computer program development facilities.
- h) Memoranda and Reports - Provided definition and direction during the project.

3.2.4.1 (continued)

In summary, the planning defined the following:

- a) Scope - The Statement of Work to be accomplished and items to be delivered was identified in accordance to the overall C&CS Program milestones.
- b) Organization - The allocation and management of the contractor's resources planned to include specific assignment of responsibility for the various tasks of the project, and thereby providing clearly defined areas of control. This organization clearly allowed for ease of communication between the task groups and the customer and provided for single-level control of each task.
- c) Task Breakdown - The definition of major task areas, the further subdivision of these areas, and the scheduling dependency relationships between tasks was the major effort in the planning phase. It provided basic milestones for the successful monitoring of the progress of the project. Manloading estimates were formulated and attention called to necessary long lead time items such as training and equipment. The Preliminary Design Review and the Critical Design Review accepted baselines were used to update these estimates.
- d) Navy Manager Support - The support provided to the many design, integration, testing, and certification tasks was defined for the Navy Software Manager. This definition was updated as further direction was available during the project.
- e) Visibility and Controls - The procedures that provided project visibility and control, and specifically, features such as design reviews, quality assurance, and configuration management were provided primarily from the project engineer's activity and used top

3.2.4.1 (continued)

e) continued

down program development processes described in section 2.

f) Facility Planning - This activity planned the facilities, hardware, and software required to support the system, controller analysis, program coding, debugging and checkout, and integration and qualification tests. These were long lead items and required early definition. This planning also included usage procedures and techniques to support procurement, modification, and development of new software to support these procedures. Other special purpose equipment and computer software was identified to support special purpose studies.

g) Training - Planning was provided for the training requirements necessary to support the new facilities and training methods.

3.2.4.2 Analysis

Before the Program 4 computer software could be developed, detailed software requirements were established for each program function. Each software requirement was described functionally and mathematically, and all interfaces were defined. This was accomplished by following the analysis plan. The analysis efforts provided many major items necessary for software development, including the following:

- a) A complete set of functional requirements for the digital controllers presented in a Computer Program Performance Specification.
- b) Performance predictions of the digital control systems, including comparisons with the corresponding continuous system and pertinent sections of the ship's specifications.

3.2.4.2 (continued)

- c) Hardware and interface requirements based on analysis of interface and hardware parameter effects on system performance and software design.
- d) Software interface requirements and effects of key interface parameters on system performance.
- e) Tradeoff study results of digital controller mechanization alternatives.
- f) Failure detection analysis and resultant software requirements.

3.2.4.2.1 Inputs to Analysis Tasks

Technical information, constraints, and guidance information were provided in documents obtained from interfacing tasks and generated during the task reviews. These inputs are identified as follows:

- a) Applicable Documents - The following listed documents provided task guidance, specifications, constraints, and some fundamental technical inputs.

3.2.4.2.1 (continued)

<u>DOCUMENT</u>	<u>MAJOR INFORMATION PROVIDED</u>
1. Program 4 Software and Systems Integration, Work Breakdown Structure.	General task description
2. Automatic Controllers, June 1973.	Detailed technical description of continuous system (analog) course and depth-keeping controllers.
3. Specifications for Building Ship, May 1973.	Specifications which must be adhered to for the Program 4 System.
4. Software Management and Performance plan for the Program 4 System.	General development plan, deliverables, schedules, and interfaces with other parts of software development.
5. Command and Control System Specifications, Sections I, XVII, and XX.	Sec I - System-level requirements that must be satisfied by the Program 4 System and interface with C&CS. Sec XVII - Type and degree of common services and support provided by the Central Computer Complex. Sec XX - Type and degree of support provided by the C&CS to the Program 4 System.
6. Contract guidance drawings.	Definition of Control System hardware configurations.

3.2.4.2.1 (cont.)

<u>DOCUMENT</u>	<u>MAJOR INFORMATION PROVIDED</u>
7. Management Development Plans	
a. Computer Program T&E Management and Technical Plans.	Shipboard and Land Based Evaluation Facility testing requirements.
b. C&SCS Computer Program Configuration Management and Quality Assurance Plans and Procedures.	Impact of Quality Assurance and Configuration Management Plans on design.
8. C&CS Interface Design Specifications.	Constraints and guidance on interfaces between Central Computer Complex and Program 4 hardware and software.
b) Interfacing tasks - Several detail design tasks were in progress at the beginning of Program 4 which provided information to the development task. Those tasks which have a major input to Program 4 development are identified below along with the major information provided.	

<u>TASK</u>	<u>MAJOR INFORMATION PROVIDED</u>
1. Control Detail Design	Continuous system controller definition and performance predictions and definition of computations required.

3.2.4.2.1 (cont.)

<u>TASK</u>	<u>MAJOR INFORMATION PROVIDED</u>
2. C&SDetail Design Tasks:	
a. Data Processing Sub-system (Signal Data Converter).	Interface definition and/or constraints.
b. Monitoring Subsystem	Definition for use in fault detection.
3. Program 4 Study and Prototype	List of signals that must be generated as software and associated algorithms for each signal.
c. Review Inputs - Reviews on the progress of technical efforts and analysis activities resulted in guidance and technical direction for software development analysis. These inputs are summarized below:	

<u>REVIEW TIME</u>	<u>INFORMATION OR GUIDANCE PROVIDED</u>
1. Program 4 development task progress meetings, CCS soft-guidelines and decisions affectware users meetings, and De-ing development. sign Working Group meetings.	Conference reports itemizing progress meetings, CCS soft-guidelines and decisions affectware users meetings, and De-ing development. sign Working Group meetings.
2. Review of Software Development Plan for Program 4.	Comments to and ultimate approval of plan.
3. Computer Program Performance Specification.	Comments to utimate approval of plan in accordance with review cycle.
4. Review of Technical Design Reports.	Technical comments and report acceptance.

3.2.4.2.2 Analysis Description

Analysis consisted of controller implementation, failure detection monitoring, reliability/availability, and manual and semi-automatic control.

- a) Controller Implementation - The real-time digital implementation requirements of the controller resulted from an iterative design process consisting of analytical design and simulations followed by performance analysis and controller modification. The general steps of this development procedure are described below.
- 1) Analytical Design - Complete descriptions of the continuous system controller were inputs to the digital controller development. The starting point consisted of reviewing these controllers and verifying their acceptability. In addition, performance data was generated or collected and used as a basis for specifying and evaluating the digital controller performance.

Digital versions of the continuous system controllers were obtained using several different state-of-the-art realizations, considering both system performance and software development requirements, were selected for each controller.

Since the total system was hybrid in nature, consisting of both analog and digital portions, the design procedure considered the effects of interface characteristics. Interfaces were modeled in terms of quantization, truncation, sampling frequency, signal delay, interchannel displacement, noise levels, etc.

3.2.4.2.2 (continued)

- 2) Simulation - Due to the complexity of the equations of motion, the hybrid nature of the system, and the effects of random disturbances, the equations were solved by computer simulations.* The simulation approach was selected to allow the many facets of the control systems software development to be studied and optimized. The approach selected allowed the interface effects to be simulated, the digital controller algorithms to be programmed, and the ship dynamics to be simulated. The simulation allowed effects of key system parameters (quantization levels) to be studied and provided data that was utilized to specify performance of the Program 4 Application Program. In addition, the simulations provided baseline data for evaluation of the operational software.
- 3) Performance Analysis - Using the simulation results describing the digital controller performance and the data describing the baseline analog controller performance, the analog and digital controllers were compared. Since several different digital controller algorithms were derived and simulated, performance differences were expected. These differences were studied along with the software requirements of each algorithm. A decision was made as to the adequacy of the digital controller's performance based on considerations of closed loop controller performance, impact on hardware and software interfaces (signal data converter and sensor requirements), and impact on software requirements (required, algorithm complexity, etc.).

* Simulations referred to here are for development of the software requirements and not for evaluation of the developed operational software.

3.2.4.2.2 (continued)

- 4) Controller Modification - Iterations in the digital controller design were planned since it was unlikely that the first design would satisfy all system performance, hardware, and software requirements. The straight forward approach of converting the continuous system controller equations to their digital counterpart might not have resulted in satisfactory closed-loop system performance due to computational delays, quantization of input signals, skewed data, finite sampling times, etc. Also, the straight-forward approach might have placed unnecessarily rigid requirements on interfaces (SDC, Digital/analog (D/A) converters). Controller modifications and tradeoffs between hardware and software implementations of portions of the controller were considered as required to meet performance goals.

Other design approaches that were based on more direct digital controller synthesis procedures, from the theory of sampled data systems and digital filter design, were considered.

- b) Failure Detection Monitoring - Ship safety and mission success were important considerations in the design of the Program 4 Application Program. Undetected failures in the Program 4 system could have lead to unsafe operating conditions and possible mission failure. It was important, therefore, that failures be detected and alarms and displays provided so that corrective action could be taken. An anlaysis was conducted of each control system, including all control subsystem components, during detail design. The software requirements for failure detection were defined under the Program 4 development task. This activity was coordinated closely with the customer.

3.2.4.2.2 (continued)

- c) Reliability/Availability - Alternative methods were analyzed for improving the reliability and availability of the control systems through software design. It should be noted that reliability analysis of Program 4 components and C&CS hardware occurred as part of other detail design tasks. The software development task concentrated on techniques for improving availability, such as automatic switching to alternate modes. The software requirements to do this were analyzed and defined and included in the performance specification (CPPS). Mode switching required simulation efforts to study effects of switching times, initialization, bad data, etc, on system performance.
- d) Manual and Semi-Automatic Control - The controller implementations discussed in 1 are concerned with the automatic modes of control and operator aids control. The Program 4 software also supported the semi-automatic and manual modes of operation by providing real-time solutions to the specified algorithms. The display/instrumentation support requirements were defined early in ship design. No additional algorithms or display outputs other than those approved in the Program 4 CPPS were provided to support the design and operation of the control station.

3.2.4.2.3 Reports

Computer Program Performance Specification (CPPS) - Program 4 was written by the customer in accordance with Weapons Specification WS-8506, Revision 1. The review cycle for the document is discussed in program 3.2.4.3.4.

3.2.4.2.3 (continued)

Analysis reports were prepared to document the development of the Program 4 software requirements presented in the CPPS. As a minimum, these reports described the basis for the software requirements of each major control function.

The reports contained traceability back to the continuous system controllers in each applicable case and included system description, significant mathematical developments, simulation descriptions, and simulation results. Major system tradeoffs were described, particularly those relating to computational and interface parameters.

The analysis reports provided the detailed technical background for the CPPS provisions and, in some cases, were reproduced in part or whole in the CPPS. Since the CPPS was scheduled for early delivery, issuance of the approved technical reports in many cases followed issuance of the CPPS. However, the contents of the reports had been discussed with the Navy during regularly scheduled progress meetings and Navy approval of technical approaches, decisions, etc. were documented with conference reports.

On Program 4, our customer was responsible for performance of the detailed analysis and generation of the CPPS and Analysis Reports. Sperry Univac received as background information all available reports and the CPPS, and performed a synthesis of this data into the software performance requirements for proper understanding of Program 4. During the period of learning, any and all questions concerning the software implementation of the CPPS were answered by telephone conversations between the project analyst/programmer and the customer's design group. The written telephone record served as control by the customer and provided good visibility at this development stage.

3.2.4.3 Design

This phase defined the design requirements of the Program 4 Application Program. The design is based on the approved functional requirements of the performance specification on the control design studies, and on the C&CS design requirements. The resulting design specification described these requirements and provided for formal control of the program design.

The performance specification testing requirements were also expanded in this phase and documented in the test plan and informal test guidelines. The test plan defined the procedures for qualification testing. The test guidelines aided in program debugging and checkout and in design of the simulation and testing functions.

3.2.4.3.1 Design Inputs

The program design was based on the requirements of the documents shown in figure 3-1. The C&CS requirements are described in the following documents provided by the CSSM:

- a) CCS Specification - Identified the integrated processing facility and the required hardware and software component specifications.
- b) CCS Design Data Document - Identified common design functions and resource allocation.
- c) CCS Software Management Plan - Defined the C&CS approach to the planning, management, control, generation, testing, integration, and acceptance of the C&CS computer programs.
- d) System Interface Design Specification - Identified the system interfaces.

3.2.4.3.1 (continued)

- e) Configuration Management and Quality Assurance Plan defined the system requirements, including computer program generation guidelines with design standards, computer program design review requirements, computer program performance testing requirements, and internal configuration control requirements.
- f) Reports and Memoranda - Provided by the interfacing tasks on overall design and by the meetings and conferences.

Other design inputs were implicit in the choices of equipment, system software, and interfaces. For example, use of the Common Program specified the interfaces and control structure of the operational program. The Program 3 operational characteristics define the signal data convertor interface.

3.2.4.3.2 Design Process

The design phase is discussed in three areas; program structure, implementation approach, and testing requirements. Each area has its own distinct outputs and characteristics.

3.2.4.3.2.1 Program Structure

Since the program structure provided the framework for completing the task, it was defined early in the Design Specification.

The program structural design was based on the multiple entry module concept of the Common Program. This program also provided general services to the user modules. In particular, it provided the major portion of the I/O services as performed by Program 3 and the Common Program message handler. Besides providing these Common Program interfaces the design provided compatibility with the other system programs including inter-

3.2.4.3.2.1 (continued)

faces and compatibility in shared computer resources including core, Central Processing Unit (CPU) time, common data definition, and common functions.

Design tradeoffs in design analysis were based on the following critical standards listed in order of importance:

- a) Optimized Program 4 system performance.
- b) Optimized C&CS performance.
- c) Optimized Program 4 software performance.
- d) Practical degraded performance.
- e) Practical development and testing characteristics.
- f) Suitability for functional changes and additions.
- g) Suitability for dedicated operation.

The functions defined in the performance specification were allocated to specific entrances in specific modules. This was done by a review and tradeoff study of the characteristics of each function, including the timing requirements, priority requirements, interrelationships, and applicable modes. The modules were formed from functions with similar entrance requirements, with consideration for multi-processing. The functions performed in each entrance of each module were described, including the logic for the function and entrance call. Also, the regional data stores were defined based on the module common data requirements.

Studies were made of the data flow and control path flow in each program structural design considered. An error analysis was made, including studies of data loss, data modification in conversions, and timing interactions. The special requirements of the Program 4 system for recovery from system-detected

3.2.4.3.2.1 (continued)

errors and from data loss due to component malfunctions were also studied. Finally, the propagation of errors through the system, whether from data loss, erroneous input, or loss of significance in the calculations, were studied. In all cases, suitable error recovery procedures were devised to minimize the effects of these errors.

During the design, the Computer Program Operators Manual was developed from expanded definitions of program control characteristics. The final details of the manual depended on completion of the coding and debug and program checkout.

During the design study, certain features were found which improved the design. Change proposals were generated to update inadequacies or to add improvements in the Performance Specification. Other features were not within the scope of the current project, but useful in a future update; these were collected for turnover with the program. Ideas on the design of subroutines were also saved and provided to aid the development efforts.

3.2.4.3.2.2 Implementation Approach - Implementation guidelines and standards were provided to support the design.

The development of specific computer program generation guidelines were critical for standardization and support of follow-on maintenance. A programmers' handbook was developed which contains all applicable development guidelines. Based on the contractor's experience and compatibility with the other system developers, these guidelines helped standardize the following programming practices and techniques:

- a) CMS-2 system usage rules and options.
- b) Common Program usage rules and options.
- c) Program usage rules and options.

3.2.4.3.2.2 (continued)

- d) Programming debugging and logical path testing.
- e) Program correction control and records.
- f) Program development historical records.
- g) Top-down structured programming usage, including progress reporting procedures.
- h) Program development protection and backup procedures.

Other management procedures were developed to provide control, visibility, and communication. These were included in the programmers handbook, and include the following:

- a) Progress reporting and development control procedures.
- b) Procedures for reporting and processing trouble reports.
- c) Procedures for project information storage and access.
- d) Procedures for collecting new design features.
- e) Maintenance of project notebooks providing collected requirements and guidelines.

The implementation of the design was monitored by the design personnel through informal review meetings. This procedure allowed the free exchange of ideas to promote the design understanding required by coding personnel.

3.2.4.3.2.3 Test Requirements

This portion of the development was a separate and parallel activity to the design of the application program. It designed the exact tests and facilities needed to support the qualification testing. The result of this design effort was the development of the WS-8506 Computer Program Test Plan. This document provided the testing requirements for qualification and required formal approval after an In-Process Review. As

3.2.4.3.2.3 (continued)

an aid to other testing, including debugging, checkout, and facility and test software development, informal test guidelines were prepared.

3.2.4.3.3 Design Outputs

The primary output of the design phase was the Design Specification. This document provided the basic structure of the developed program and guidance on the development of this program. These requirements, which guide further development were approved by the C&CS as described in the next phase.

The design phase also developed other documents which, except for the informal test guidelines, were approved and maintained through the In-Process Reviews (IPR). These documents were:

- a) Computer Program Test Plan.
- b) Computer Program Operators Manual.
- c) Test guidelines.

3.2.4.3.4 Design Reviews

Upon Sperry Univac's delivery of the preliminary Computer Program Design Specification (CPDS), the Navy held a formal critical Design Review (CDR) based upon detailed questions and comments from the customer's design analysis group and other Navy contracted development agencies affected by interfaces in Program 4. The documented minutes of this CDR and the results of testing the Level 1 and 2 program were accounted for in the approved final CPDS.

3.2.4.4 Code and Debug

The program code and debug phase of the Program 4 computer software development was concerned with the actual implementation of the approved design through the development and debugging of

3.2.4.4 (continued)

the individual functional sections of the code. The coding and debugging process was controlled by the program generation guidelines and other management procedures.

3.2.4.4.1 Inputs to Code and Debug

The following provided support in coding and debugging:

- a) Design Specification - The CPDS provided the allocated baseline requirements.
- b) Performance Specification - The CPPS provided the system functional requirements.
- c) CMS-2 System Description and Associated Guidelines - These described the CMS-2 language implementation and features, and provide usage guidelines.
- d) Compile Facility Description and Operating Procedures - This document was provided by the Navy and described the available development facilities at the Land-Based Evaluation Facility.
- e) Common Program Description and Programmers Reference Manual - These manuals were provided by the Navy and provided details for interfacing with the Common Program and utilization of the available system services.
- f) Program Generation Guidelines - This document standardized coding requirements and use of structured programming.
- g) Test Guidelines - These informal guidelines were input from the design phase and provided direction to the debugging.

3.2.4.4.1 (continued)

- h) Test Plan - This document was partially developed during the design phase and indicated the qualification testing requirements. It also aided programmer development of debugging requirements.

3.2.4.4.2 Program Coding And Debug Development

Sperry Univac produced, tested and delivered Program 4 in four iterations called level 1, level 2, partial and final. In level 1 all interfaces to software outside the application program were emulated and computer resource consumers of CPU time and computer memory were modeled in the code. In level 2 interface between program modules of the subprogram was coded. The partial program was a complete program as determined from the specified information. The final program was an update to the partial program and was completed to the customer's satisfaction.

Program 4 coding consisted of generating CMS-2 language statements suitable for compiling and generating executable computer programs, and for documenting coding details, corresponding operational functions, and overall program structure. The developed code conformed to the restrictions imposed by the developed programming guidelines as described in paragraph 3.2.4.3.

The coding tasks were scheduled so that high-risk items and highly interdependent functions were developed first. This also supported the intermediate program (level 1, 2, and partial) delivery requirements. This included consideration of the development of those functions which interface with the other C&CS subsystems to provide early interface validation. The later developed items then used these early items in their development.

3.2.4.4.2 (continued)

Adequate provisions were made to ensure quality development. The available program librarian features were used to record the current state of each coding task. The purpose of each compiler job submitted was described by run stream comments. This information on past runs was maintained in a historical file to provide access to past efforts.

To provide assurance of a practical backup position in case of catastrophic system failures causing loss of generated source libraries, strict library usage rules were generated and observed. This included the preservation of several generations of library tapes with corresponding system tapes and program listings.

Debugging consisted of following procedures which verified that the executable code of each module functioned properly.

Programmers developed and followed an informal debug plan. Much of the debugging took place at the compile facility using "load and go" features of the CMS-2 system, with data driver programs providing interfaces with the coding. This type of debugging was easily documented by the system output listings and enabled close monitoring.

3.2.4.4.3 Outputs from Code and Debug

The program code and debug phase ended when the debugged coding was placed in the overall program structure for further checkout. This was an iterative process, with changes developed during checkout going through debug before checkout each time.

3.2.4.5 Checkout Testing And Integration

In this phase the subprogram packages were checked out using the Common Program, Program 3, and the Simulation and Test Program. The first package verified was the Regional Control

3.2.4.5 (continued)

Module which served as the Program 4 local executive routine. New packages were checked out and then integrated into the existing program, and the total system integrity was reverified. This procedure was capable of producing the intermediate program deliveries necessary to support the system/subsystem software integration process at the land-based evaluation facility (LBEF). The result of the program checkout and integration phase was an application program ready for formal qualification testing.

3.2.4.5.1 Inputs to Checkout Testing And Integration

The following internally produced documents were required:

- a) Test Plan.
- b) Test Procedures.
- c) Scenarios.
- d) Simulation and test program operation and procedures.

The software required included the following:

- a) Common Program.
- b) Program 3.
- c) Simulation and Test Program.
- d) Subprogram packages.

3.2.4.5.2 Checkout Testing And Integration Procedure

Before the qualified Common Program was available, a preliminary version was used, supplemented by simulations of missing functions as required. The scheduled availability of the qualified Common Program was factored into the checkout phase to allow maximum usage.

3.2.4.5.2 (continued)

The schedule availability of Program 3 required the development of routines with those equivalent functions needed during checkout. Final checkout procedures required use of the qualified Program 3.

Checkout of the subprogram packages (program modules) required extensive use of the Simulation and Test Program. This provided the realistic interaction between the computer software and the simulated external environment and possible data paths. The simulations developed are described in Section 4. In order to provide repeatable checkout situations, the simulation was controlled by a scenario input.

The basis for measuring the performance of some of the checkout tests was data collected during the analysis phase of development. Some checkout tests required rerunning the special purposes simulation programs to obtain test input data. Test input data was also generated from consideration of the requirements of the performance and interface design specifications. All tests required were specified in the approved Test Plan and were conducted in the manner detailed in the approved Test Procedure document.

Testing and integration were performed in stages. The Regional Control Module was verified first since it was the local executive routine for Program 4 computer software and handled all communication with the Common Program and Program 3. Within each module or sub-program, all interfacing paths to other programs, interfacing equipment, and operators were tested. All internal logical paths were tested to ensure that timing and sequencing performance requirements were satisfied. New subprogram packages were tested and then integrated into the existing version of the program. This new version was reverified to ensure total system integrity.

3.2.4.5.2 (continued)

The qualification tests were performed during the testing phase in an informal manner, as possible. This verified proper operation and functioning of the qualification tests. Most checkout testing used the general purpose Simulation and Test Program, but some testing required the development of special purpose drivers or the modification of the Simulation and Test Program to provide needed capabilities.

An important aim in this local integration was to prove that proper operational control was provided by the Program 4 computer software. One aspect of this was to test the utility of the displays and controls on the signal data converter with the application program running under the Common Program and Program 3. Due to the condensed schedule and the unavailability of the actual station, early testing was done at the software developer's site, utilizing the prototype station. It should be noted that qualification testing did not require the prototype station, hence its delivery date did not affect the schedule but could affect quality assurance of the computer software.

3.2.4.5.3 Outputs from Checkout Testing And Integration

The application program package ready for qualification testing was the most important output of this phase. The customer was notified of the estimated completion in sufficient time to properly schedule the formal qualification testing.

Partial program deliveries allowed early LBEF testing of system utilization and interface verification. These deliveries were the Level 1, Level 2, and Partial Program packages. Each was a check-out operational program representing the then available functions from the top-down programming development effort. Each was checked out using the Common Program and a test driver. The delivery was also accompanied by support personnel to ensure the proper functioning of the module in the LBEF environment.

3.2.4.6 Qualification Testing

This phase of testing demonstrated that Program 4 completely satisfied those requirements levied by the performance, design, and interface design specifications. The detailed test requirements were defined in the test plans and test procedure documents, and were supported by special purpose test software. The result was the establishment of the program product baseline which entered C&CS configuration management and was available to the Navy for further testing.

3.2.4.6.1 Qualification Testing Inputs

The following are inputs for qualification testing:

- a) Program Package - The operational program package developed during the checkout phase was the basic input. This was validated to the contractor's satisfaction, including performing the Test Plan testing. This program was integrated with the qualified Common Program, qualified Program 3, and suitable dummy programs to simulate the other interacting subsystems.
- b) Performance Specification - The CPPS provided a functional description of the performance of Program 4 software and was the authority for evaluation of test results on system performance.
- c) Design Specification - The design specification contained the design description of Program 4 and was the authority for evaluating all design test results.
- d) Test Plan - The Test Plan described the specific tests to be performed during qualification for satisfaction of performance and design requirements.
- e) Test Procedures - The Test Procedure described the performance of each test, delineating responsibilities and actions.

3.2.4.6.2 Development of Qualification Tests

Qualification testing formed the basis for approval of the delivered Program 4 and provided assurance that the program met all performance and design requirements. The elements in this testing included the following:

- a) Detailed qualification test plan and test procedures approved by the customer and the Navy.
- b) Special purpose computer programs allowing simulation of the application program and measurement of the performance.
- c) The customer assigned review team to witness and sign-off the successful completion of the formal test procedures.

A special purpose test computer program provided the customer with assurance of proper simulation to test the required functions properly. This computer program was provided to the integration group at the LBEF to ensure proper functioning of all components in that environment. The program is described in paragraph 3.2.4.9.

Scheduling of the qualification tests was directed by the customer and his assigned review team. Each test was run as described in the test procedures; the results were reviewed and successful completion noted. Reruns were made, as required to provide assurance of results.

Upon satisfactory completion of the qualification exercise, the C&CS program product baseline will be established and the program will be delivered to the Navy for integration testing and eventual C&CS certification at the land-based evaluation facility. At this time, the program came under Navy configuration management procedures and the customer assumed a support file for further testing of the Application Program.

3.2.4.6.3 Output from Qualified Testing

Qualified testing resulted in delivery of the final program package to the Navy. The delivered package included copies of the error-free source libraries and of the same application program integrated with the Common Program as tested during qualification. Sufficient test programs were included with this program, as required, to ensure valid operation at the LBEF. Other deliverable items were included, as describe in NAVORD WS-8506 and the C&CS Software Development Plan, for formal delivery of the computer program package. Contractor support was supplied, as requested, to assist in the initial validation of the program.

3.2.4.7 Qualification Review Support

The qualification review was performed at the LBEF by the Navy as describe in the C&CS Software Test and Integration Management Plan. The objective was to establish that the Application Program was compatible with the LBEF computer system. Detailed testing of the review was assumed to be a subset of the qualification tests. The customer assisted, within the constraints of the established support level of effort, in these tests and in developing and conducting additional tests required by the Navy. Part of the support of this phase was an early description of the nature of the testing to allow proper LBEF preparation and scheduling. Early checkout of special purpose test software was required to expedite the later tests.

3.2.4.8 LBEF Support

At the completion of the qualification test for the Application Program, the contractor entered a support role. In this phase, he aided the Navy as required, in support of the Test, Evaluation and Integration (TE&I) activities performed by the Navy at the LBEF. These activities lead to certification of the C&CS program and the commencement of shipboard testing.

LBEF TE&I support started early in the program to assist in the LBEF development. The contractor provided the LBEF with requested information in support of the facility design. The testing requirements, particularly testing of interfaces and non-degraded Program 4 performance, was supported as required. This included aid in development of operationally significant test scenarios. An important item to the support was the Level 1, Level 2, and Partial Program deliveries.

The customer provided personnel, as required, in support of LBEF activities. This included the loading and initialization of programs and the operation of the Program 4 manned stations. The customer had no equipment at the LBEF and no maintenance functions.

Program checkout included the location and diagnosis of subsystem computer program troubles. Special purpose tests and test programs were developed and executed as required.

Program changes included the generation, as required, of Field Change Notices (FCNs) in conformance with the Navy Configuration Management Plan and Procedures. Integration agency monitored all C&CS ECPs for their effect on Program 4. These changes may have required the development and performance or re-performance of qualification tests in order to ensure the re-qualification of the product baseline. The contractor also updated required NAVORD WS-8506 documentation to reflect the changing baseline.

3.2.4.9 Shipboard Qualification Support

The Navy was supported by the Program 4 software contractor during the shipboard installation, operational evaluation, and sea trials. This support included early design assistance and direct personnel assistance.

3.2.4.9 (continued)

Early design assistance was very similar to the support provided to the LBEF. Specifically requested information on the characteristic and testing requirements of Program 4 were provided to support the design and scheduling of the shipboard testing. Of particular interest was the interface testing requirements.

Direct personnel support included the functions discussed in paragraph 3.2.4.8 reapplied for shipboard testing. This support was an application of Sperry Univac's experience with the program and the ship's systems. Direct support provided quick and accurate diagnosing program and interface troubles and provided corrective action, as requested by the Navy.

3.2.4.10 Simulation And Test Program Development

To ensure objectivity, the simulation and test program were developed independently of the operational program. Both the development plan from the planning phase and the performance specification from the analysis phase of the application program development provided initial requirement definition for the simulation and test program development. The ship design documents also provided necessary information on the Program 4 system description. The method of program development closely paralleled that of the operational program with the exception of the design review phases. Informal program documents were used to establish performance and design specifications for the simulation and test program. These informal specifications provided on-time delivery to support LBEF simulation and test design efforts. Also, these informal specification documents allowed a simplified revision of the simulation and test program for use as the Level 2 simulation program.

3.2.4.10 (continued)

The simulation and test program was tested by comparing the performance with existing simulations on other machines. A formal test plan was not utilized. The verified simulation and test program provided the basis for checkout, integration, and qualification testing of the operational program.

The simulation functions realistically duplicated the control system's characteristics, the environment, and the hardware and software characteristics of the C&CS central computer complex. The simulation was controlled by scenario inputs and operator inputs, and by the commands generated by the operational program. Figure 3-6 shows the functional relationship of the simulation functions and test program with the Common Program and Program 4.

The test program was developed utilizing the top-down structured programming approach. The capability to control individual functioning elements was developed first; this was used in the checkout of each subprogram package (program module).

An open loop simulation test function was developed based on scenario inputs from off-line simulation programs previously referenced. These scenario inputs included the following information:

- a) Exact Signal Data Converter input data words at the input sampling rate.
- b) Control commands to simulated interfaces, including the Signal Data Converter, disk, magnetic tape, and computer hardware.
- c) Test control.
- d) Explanatory comments.

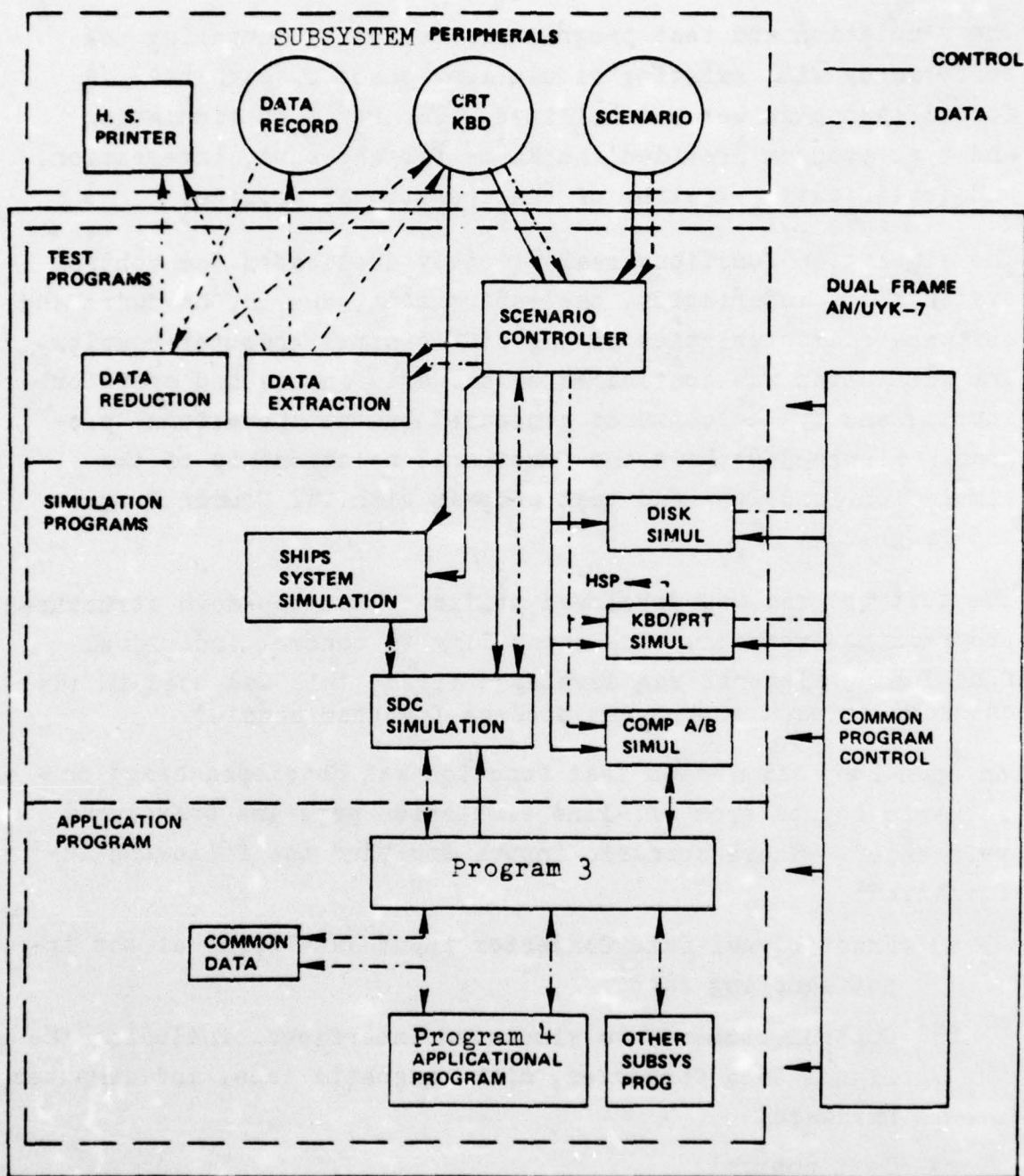


Figure 3-6. Simulation and Testing Functional Interfaces

3.2.4.10 (continued)

The resultant outputs of the Program 4 Application Program were compared with the previously obtained verified results to assess performance. These were also used to provide simple generation of equipment failure characteristics.

The closed loop simulation test functions, based on scenario input, controlled the operation of the various functional simulations, thereby allowing the application program outputs to control specified parameters. The signal data converter, disk, magnetic tape, and computer hardware simulations were augmented by control functional simulations, including the station position. Versions of this closed loop test program were used as the level 1 and level 2 simulation programs. Mode switch-over characteristics were added to obtain the partial simulation program.

Data extraction, data recording, and post data reduction routines were utilized to analyze test results. Some of these programs were currently available as part of other Government developments, and were utilized if feasible. The development of any new routines was coordinated with other activities having similar requirements.

3.2.4.11 Operators Manuals

Following delivery of the level 2 and final program packages, the Sperry Univac Program 4 development group produced the Computer Program Operators Manual (CPOM) in accordance to the NAVORD documentation standard. WS-8506, Revision 1. This operators manual was fully descriptive of the operation of the then completed Program 4 software. The CPOM served an important role for training of personnel operating the system and was necessary for defining proper resolution for correct operation and recovery from error conditions.

3.2.4.12 Configuration Management

Configuration management was applied to the Program 4 Application Program and was maintained through its life cycle to:

- a) Assure proper identification of the functional and design specifications of the program baseline.
- b) Control the identification and changes to those specifications.
- c) Provide a means for processing changes, recording change processing, implementing changes, and recording change implementation status.

The Navy developed C&CS Configuration Management (CM) Plan was involved during the top down development and continued maintenance of the baseline of Program 4. All Program 4 delivered items (program specifications, packaged software, test plan, test procedures, manuals, and subprogram design documents) of both the application program and the simulation/test program were placed under formal configuration control upon acceptance by the Navy. This formally applied configuration management is more fully described in section 6.

Throughout development, Sperry Univac maintained configuration management of the leveled baselines of Program 4. As already described, the development of the program specification, design and test plan was controlled through informal communications and formal review/approval by the procuring agency and by the Navy. Through these reviews, the product baseline was established. Technical management of the development group assured proper generation of the determined operational functions of each leveled delivery of the program package.

3.2.4.12 (continued)

CM group in Sperry Univac and the procurement agency formally checked for compliance to the determined specified baseline and provided the acceptance criteria of each delivered item. As each item was moved from development to review, acceptance and delivery it was maintained in the Program 4 project library. Control and location of items in this library were controlled by an automated catalog file maintained on the Univac 1108 using a program called MAPPER.

An integral part of the total CM on Program 4 was the control of program code. Configuration control of code started when a programmer released a debugged subprogram (program module) signified by signing a Release Notice for checkout testing and integration into the developing baseline. The magnetic tape record of the compiler directives, source code object program listing and test procedures were placed together with the signed Release Notice into the project library. The only software accepted in the evaluation and integration test was that under configuration control and maintained in the project library. Whenever it was necessary to make a change to a controlled program, the following steps occurred:

- a) The development organization documented the problem on a Computer Program Problem Report (CPPR) and prepared a Computer Program Change Order (CPCO) including a Program Maintenance Log showing the object change (patch), the corrective source library change code, and affected program listing.
- b) The development organization manager signed the CPCO indicating approval and submission to the project library.
- c) The project librarian updated the index, program listing and documents affected by the change.

3.2.4.12 (continued)

Upon satisfaction of the checkout testing of each program module, the baseline master file was updated in the project library in accordance to the approved PCCOs. The master copy of the computer program package and documentation was retained in the project library. The librarian distributed copies from this baseline master to locations for deliveries and other activities as directed by the procurement agency.

After delivery to the customer the delivered items remained as controlled items in the project library. The responsibility for CM was shifted to the customer and the Navy in accordance to the Navy provided C&CS Management Plan. Sperry Univac supported the customer throughout qualification testing. C&CS system evaluation and integration at the LBEF and shipboard system qualification. As part of this support, the project library served as the local point for receiving, logging and transmitting all STRs and ECPs. After approval by the customer for resultant program change actions, Specification Change Notices (SCNs) and Computer Program Change Orders (CPCOs), the project library received updates to the master data and provided for transmission of copies to all approved recipients.

SECTION 4

PROGRAM TESTING (Task 2.3)

4.0 INTRODUCTION

This section describes the program testing practices employed by Univac for each of the four programs under study. Program testing methods employed on the programs are classified according to conventional program testing or top-down concept program testing.

The description of conventional program testing, as used for Program 1 and Program 2, specifies in detail the approaches and considerations for the testing of a program as a "completed" isolated building block.

The description of top down concept program testing, as used for Program 3 and Program 4, specifies in detail the approaches and considerations for the testing of a program as an "evolving" isolated building block.

Within each method the subjects considered were: testing objectives, responsibilities, scope and extent, levels of testing, test system viability, design for ease of testing, methods and tools used, and testing documentation generated. Other topics threaded through the study include evolving techniques, advantages and disadvantages of the testing methods, support tools developed, and carry-over of knowledge from project to project.

4.1 CONVENTIONAL PROGRAM TESTING

In the usual simplified model of the traditional program development approach there are four phases of work: analysis, design, production, and testing. These phases are time-serial with respect to each other.

4.1 (continued)

During the fourth phase the program was tested to assure that all performance requirements were satisfied and that the program was operationally error free since almost the entire program was available to be tested at one time. The usual testing method was bottom-up. That is, the independent functions were tested first followed by the dependent. Certification proceeded in a building block manner with those functions certified first, being used to certify the remaining functions.

However, program testing did not start in practice from a fixed point in time. Program testing may have been initiated several times only to find that the checkout, or even design objectives, had not been completed. Testing effectiveness was normally diminished at first by unexpected interactions between the large number of program elements including the test tools which were not yet executed as an integrated software system. By the time good testing progress was being made, most of the development schedule may have been consumed. As a result, program testing personnel were being asked to complete their work in insufficient time. Under the pressure of insufficient time, any technical problem may result in some question of the test integrity. For example, testing discoveries of program inadequacies may have been recorded with insufficient data to clearly define them. Therefore, testing could be questioned concerning test validity, the source of the problem, personnel competence, etc. These pressures would cause additional management problems for control of the project and for visibility to the customer. When testing was performed as a one-shot effort following completion of production, this burden seemed to be inescapable.

This conventional testing approach should be compared with the advanced top-down techniques employed by Univac on Programs 3 and 4.

4.1.1 Program 1

Program 1 consisted of an operating system program, a navigation application program and a simulation program as described in the Program Environments section of this document. The simulation program provided wrap around simulation of the operational environment normally encountered by the navigation system and was used as an aid in the development and certification of the navigation program.

Developmental testing, program certification, and software integration were all provided by Sperry Univac for Program 1. Program testing performed and criteria satisfied are discussed in the following subparagraphs.

4.1.1.1 Testing Objectives

Testing objectives for Program 1 were:

- a) Certification of the Operating System, Navigation Program, and Simulation Program during program development.
- b) Operational testing of the Navigation Program.
- c) System integration testing performed by Sperry Univac including:
 - a) Land based operational tests using full simulation.
 - b) Support during shipboard operational evaluation.

Quality assurance provisions were required in the initial system performance specifications and were expanded in the navigation program performance specifications. These formed the basic performance and certification requirements used to generate the navigation program certification plan. The certification test and a favorable test report satisfied the requirements for acceptance of the Navigation Program.

4.1.1.1 (continued)

In general, certification testing required that the independent functions be tested first, then the dependent. Certification proceeded in a building block manner with those functions certified first, being used to certify the remaining functions. Since the simulation module was required for functional certification, it's independent functions were among the first to be tested. All functional certification tests were required by the performance specification to be repeatable. These software testing objectives were determined and documented by an independent group in Sperry Univac.

All testing required by the certification tests was performed much more exhaustively during program development testing. These tests were performed on the operating system, the navigation and the simulation programs.

4.1.1.2 Responsibility

Sperry Univac had the following testing responsibilities for Program 1:

- a) Certification of the Common Program Operating System, Navigation Program and Simulation Program.
- b) Develop the necessary test software.
- c) Write the Computer Program Test Plan, Computer Program Test Procedures and certification test reports.
- d) Provide support during operational testing of the Navigation Program.
- e) Perform integration of the C&CS operational program operating system and test it for operability.

4.1.1.3 Scope and Extent

Program 1 testing verified that Program 1 software satisfied all functions specified in their respective program performance and program design specifications. The formal certification test procedures were the basis for official acceptance by the Navy.

The Computer Program Performance Specifications for the Navigation Program were generated by the customer with advice and support from Sperry Univac. Sperry Univac then expanded the Quality Assurance sections into certification test plans and test procedures.

The sequence of testing for all programs was as follows:

- 1) Each programmer at Sperry Univac verified that his program satisfied an informal software checkout test.
- 2) The program was then turned over to the Sperry Univac test group who verified via a preliminary certification procedure and simulation that the software satisfied contractual requirements as specified by the Program Performance and Program Design Specifications.
- 3) The Navigation Program was delivered to the customer for formal certification testing. Sperry Univac provided support for this testing. The program was accepted by the Navy, who then turned it back to Sperry Univac for system integration.
- 4) The baseline Common Program Operating System was furnished by the Navy and modified as needed by Sperry Univac to provide the C&CS Operational Program. Test programs developed by Sperry Univac for earlier Navy programs were adapted for testing by C&CS. After intensive informal testing, Sperry Univac performed the formal certification tests for the customer.

4.1.1.3 (continued)

- 5) Software Systems Integration of the C&CS, Navigation Program, and other applications were performed by Sperry Univac, first at St. Paul using simulation of all missing hardware, and then at the Navy's Land-Based Evaluation Facility (LBEF) using live and simulated equipment. The LBEF tests featured full load endurance tests for periods ranging from 96 to 120 hours.
- 6) Sperry Univac furnished support for actual shipboard trials of the delivered software.

4.1.1.4 Levels of Testing

The Program 1 Navigation Program was tested as an independent unit. A low level functional test was applied separately to each program module function. A high level functional test was then applied to the overall program taken as a unit, and separately to each subprogram module of the program.

Testing of the Common Program consisted of unit (Static) and dynamic testing of the eight functional modules. The Common Program was tested first, followed by the simulation and application programs. Once each subprogram of the C&CS Operational Program was tested as an independent unit, Sperry Univac performed extensive testing of the integrated software system.

4.1.1.5 Test System Viability

The test control and simulation programs were designed and developed for Program 1. The test programs for the Common Program, the Navigation Program and the Simulation Programs all employed modular construction which allowed independent testing and made the programs amenable to operation on varied hardware configurations.

4.1.1.5 (continued)

The operating system employed Common Program test modules operated from test drivers with internally supplied parameters. The Navigation and Simulation Programs, however, began using externally supplied scenario control for automated-repeatable and easily-altered testing. Scenario control was utilized in later projects, including system integration testing of Program 1 and testing applied to Programs 2, 3 and 4.

4.1.1.6 Design for Ease of Testing

The use of scenario controlled tests, test driver modules, simulation of missing elements, and data extraction/reduction tools facilitated testing of Program 1 Software. The test driver modules were part of an evolving test capability used on several generations of the Common Program. The scenario controlled tests were used to automate testing of the Navigation Program, while data extraction and reduction were used extensively for testing of both itself and the Navigation Program. The simulation program could:

- a) Replace by simulation, all or part of the navigation equipment.
- b) Be resident in either the same computer as the Navigation Program or, in a different computer.
- c) Share the same executive as that used by the Navigation Program if both were resident in the same computer, or a copy of the same executive program if resident in a different computer.
- d) Be controlled by reproducible segmented scenarios from paper tape or disk. These could be overridden from the keyboard/printer.

4.1.1.6 (continued)

- e) Simulate navigation capabilities while checking out live control consoles.
- f) Be used for both program test and later system integration. This simulation capability was later extended for use in crew training.
- g) Via the keyboard/printer, provide on-line test results for instant evaluation of the navigation system.

The programs were designed with modular construction for easy expandability and ease of maintenance.

These capabilities designed into the system eased generation of the test programs, and testing of the developed software.

4.1.1.7 Methods and Tools

The formal test plan, certification procedure, and test result documents for the Navigation Program were prepared by the customer. However, the draft information used for these purposes was taken from informal test documents prepared by Sperry Univac for its own testing of the applicable software, prior to the acceptance tests.

Initially, the following testing procedure was used for the application and simulation programs:

- 1) Punched cards containing the desired testing commands, were read into a test scenario generator.
- 2) The generated scenarios were written to paper tape.
- 3) The paper tape was read into the system via the keyboard printer station to provide the system test run.
- 4) The test results were printed on the keyboard/printer.
- 5) The test results were analyzed visually and were transferred to a test report log.

4.1.1.7 (continued)

- 6) Any inaccuracies shown by testing were recorded together with test result data to a site problem log. This log was kept at the site to record any problems until they could be resolved.

Because of the difficulties involved with using paper tapes for large tests, it was found advantageous to transfer the tests to disk storage. Scenarios were readily available and easily modified. Disk storage avoided the generation, storage, and reading problem inherent in paper tape processing.

Scenarios were not employed in testing the Common Program (CP). Special test driver modules were written which tested particular functions when manually initiated. The CP modules were tested independently, then integrated and their interactions and operation tested in a simulated real-time environment.

In performing system integration, the capability was available to interchange real and simulated data input by recabing. The results showed the simulated input equaled the live data. Testing of the simulation program was performed using large quantities of sample data provided by the customer.

In general, program checkout plans, checkout procedures and checkout logs were informal. The programmer would generate them in note books for his own convenience while debugging his program.

A Navigation Program Simulator was generated and used by Sperry Univac to test the Navigation Application Program. This simulation program was generated in response to a card image input test scenario and operated in the same multiprocessor, multi-

4.1.1.7 (continued)

programmed environment as the navigation program. It simulated input into, and extracted output from the Navigation Program. This provided an easily controlled operational simulation of the varying conditions to be expected by the operational program.

Two important capabilities were perfected in developing the Navigation Simulation Program.

- 1) The program was capable of driving and extracting data from the driven program in two modes:
 - a) By internally simulated I/O using shared computer memory.
 - b) By using externally simulated I/O via end around I/O cabling, or remote computer.
- 2) Time compression was used to decrease the time between scheduled test events by incrementing the internal system real-time clock.

4.1.1.8 Test Documentation

Test documentation for Program 1 consisted of test documentation for the Common Program and test documentation for the Navigation Program.

4.1.1.8.1 Test Documentation for the Common Program

The following test documentation was prepared:

- 1) Program Performance Specification for the operating system program for Program 1, section 4.
- 2) Certification Test Plan for Common Program.

4.1.1.8.1 (continued)

- 3) Certification Test Procedure for Common Program.

4.1.1.8.2 Test Documentation for the Navigation Program

The following test documentation was prepared:

- 1) Test Plan - This document was prepared by the customer.
- 2) Certification Procedure - This document was prepared by the customer.
- 3) Certification Test Report - This document was prepared by the customer.

Informal documents generated by Sperry Univac for internal testing formed the basis for these formal documents.

4.1.2 Program 2

Program 2 consisted of:

- 1) A Common Program (CP) that was modified to become the Command and Control System (C&CS) executive system.
- 2) Simulation software for intermodule interfaces, computer peripherals, and the operational environment (i.e., wrap-around simulation).
- 3) Test software to test all software elements including the operating system and application programs.
- 4) The C&CS application programs.

In addition to the test software above, Sperry Univac generated test software and testing requirements for a Sensor System program which was written by a separate Navy contracted agency.

Program testing performed and criteria satisfied are discussed in the following paragraphs.

4.1.2.1 Test Objectives

Quality Assurance provisions for Program 2 required that test plans and procedures be generated for thorough testing of the developed software. In addition, a discipline was applied to all levels of the program's development effort, from its initial specification to its final certifications, in order to ensure a reliable product.

The testing objectives ensured that the Command and Control System (C&CS) programs satisfied the reliability requirements of real-time programs, that they met the stated performance/design requirements, and that they were structured for easier maintenance and reduced life cycle cost.

The test plans and procedures were written and examined to assure the necessary and sufficient correspondence between the tests performed and the performance requirements stated in the program performance specifications.

Guidelines were generated to:

- a) Establish standards that enhance program reliability.
- b) Permit quality to be built into the design.
- c) Reduce program maintenance problems.
- d) Provide procedures for management of program generation including the use of program libraries.

Key objectives were:

- a) The evolution of software tests into tools which are viable from project to project.
- b) Automating software tests, such as the use of scripted scenarios for control and repeatability.

4.1.2.1 (continued)

- c) To improve certification procedures for greater clarity and explanation of the tests being conducted and results expected.

4.1.2.2 Responsibility

For Program 2, Sperry Univac was responsible for producing the following program testing items:

- a) Development of a simulation program to test the C&CS intercomputer and peripheral interfaces and to provide a personnel training aid.
- b) Development of test plans stating what was to be tested. This included independent tests for the C&CS Operational Program and the Sensor System Program and a full load endurance test for the combined system.
- c) Development of Certification Procedures for:
 - 1) Common Program to energize test driver modules.
 - 2) Phase 1 independent test of the C&CS Operational Program.
 - 3) Phase 1 independent test of the Sensor System Program.
 - 4) Phase 2 full load endurance test of combined C&CS Operational Program and Sensor System program.
- d) Performance of verification testing by customer representatives.
- e) Support the customer's representatives as they performed certification testing using the procedures of item c).

Program 2 software development and Program 2 test software development were the responsibilities of two separate organizations.

4.1.2.2 (continued)

Performance specifications generated by the software development group, were used by the test software development group to produce test modules, simulation programs, test plan, and certification procedures. The software test group worked with the software development group during the integration phase to:

- a) Verify the test procedures.
- b) Debug the intermodule and inter-computer/peripheral interfaces.

The customer was responsible for review and approval of all design and documentation, and for providing personnel responsible for performing the certification tests. The software was accepted when the approved design of the program operated correctly when tested using certification procedures written to demonstrate each requirement stated in the performance and design specifications.

NAVMAT Documentation standards were adhered to for all Program 2 test documentation (refer to the section on Development Standards for Program 2).

4.1.2.3 Scope and Extent

It was verified by Sperry Univac and certified by the customer that all code for specified functions was executed under example conditions to assure a documented acceptability. These tests satisfied the requirements of the program performance and design specifications.

Testing was provided by the Sperry Univac development and test groups, and by the customer during certification testing.

4.1.2.4 Levels of Testing

Each program received a high level functional test both as a unit and as part of the integrated operating system. In the functional certification tests, an attempt was made to follow the sequence of requirements from the program performance specification, to make obvious by the actions and resulting observations that these requirements were fulfilled. Because this was done at a high level, redundant testing of the same functions at the subprogram functional level in most cases was unnecessary. Also, functions were not limited to single modules so it would have been difficult to apply a functional test to a module. All certification tests were performed in an integrated configuration with operational program modules and interfaces active. Peripheral interfaces supplied by software were provided either by live equipment or by simulation programs.

4.1.2.5 Test System Viability

The simulation program was originated on this project. This program featured modular construction and scenario control. Because of the determined design, it was amendable to many different hardware configurations and could easily be expanded to simulate additional peripheral equipments.

The dynamic "scripted scenario" test capability combined with the simulation program was the basis for the crew training program capability.

Certification tests and documents for the Program 2 Common Program were adapted from the Program 1 Common Program certification tests and documents.

Use of the common test modules and scenario controlled testing contributed to reliable long-term maintenance support.

4.1.2.6 Design for Ease of Testing

Use of simulation programs for testing simplified system debugging by isolating any software errors to specific program modules. The system could be configured in numerous ways with simulation of intermodule interfaces, computer peripherals, and operational environment. C&CS full load endurance testing was run using simultaneous simulations of three weapons computers interfaces. A derivative of the C&CS operating system software was used to control each simulator.

In general, the design of the Program 2 Simulation Program made it possible to place a controlled load upon the operational program from live or simulated auxiliary programs. Any C&CS action could be taken with the operational program, the software reactions traced, and recorded automatically.

A "scripted scenario" generation capability was used to record on magnetic tape any or all display console entries made by the operators at the selected consoles. The resulting tape then provided a dynamic tool to reproduce any selected sequence of events.

4.1.2.7 Methods and Tools

The formal test plans, certification procedures, and integration tests were generated by Sperry Univac for all software provided for Program 2. A section was generated in a test plan for each of the requirements included in the approved Program Performance Specification and cross referenced to it. The test plan then, stated what was to be tested. The high level certification procedures were generated based on these test plans and verified by testing the software against them. Program 2 test scenarios were generated manually by an operator at the simulation control console. These were usually ordered so the flow of console actions followed were close to the "real" operators normal sequence of actions for each function.

4.1.2.7 (continued)

This "scripted" test scenario function was used to give the simulation console operator the option of recording his entered scenarios on magnetic tape. It had the capability to include in the recording any or all keyboard entries made by the display operators and the times at which they occurred. Once generated, this tape could be run into the system again to automatically repeat the original test, either at the original timing, or by a specified time elapsed sequence. This provided a sophisticated dynamic testing and valuable crew training and evaluation tool.

The Common Program certification test software was developed from that used in Program 1. Common Program certification testing was accomplished by the building block method (independent functions within program modules were tested and then used with additional test software to test all dependent functions).

Preceding a test sequence, a descriptive paragraph described the test to be made and explained briefly the rationale behind the test. The "Actions" included parenthetical explanations and text references where necessary to clarify the actions taken. "Observations" were simplified to clarify interpretation of whether the test had "passed" or "failed". In some cases observed test results were analyzed by comparing them to explicit data analysis guidelines provided by the customer and included in appendices of the certification procedures.

Test results were extracted by means of:

- a) Recording the output to display consoles.
- b) Using the data extraction program, selected data pertaining to system tracks, operator events, and/or input/output buffers from the C&CS operational pro-

4.1.2.7 (continued)

gram were recorded a magnetic tape. A data reduction program reduced this data to a format selected by the operator and listed it on a high speed printer.

- c) Using capabilities (e.g., Snaps, Traces, etc.) provided by the debug aids.

For tests "failed", a STR was generated describing the deficiency in detail and included supporting dumps and descriptions. Resolution of the deficiency involved one or more of the following:

- a) If a software problem, generate errata, verify and deliver the corrected code, for a solution.
- b) If a procedural or documentation problem, generate a temporary correction or addition.
- c) If a change affected a Navy controlled document, to make a permanent correction to the problem an Engineering Change Proposal (ECP) was generated. When approved by the Navy Control Board, the permanent solution was generated, verified, and delivered as the modified code documentation change package.

Program checkout plans, checkout procedures, and checkout logs were informal but required to be maintained by every programmer.

4.1.2.8 Testing Documentation

The following test documents were developed by Sperry Univac for Program 2:

- a) Certification procedures for the Common Program Operating System.
- b) A Phase I unit test plan for the C&CS operational program.

4.1.2.8 Testing Documentation (continued)

- c) A Phase I Certification Procedure for the C&CS operational program.
- d) A Phase I Certification Procedure for the sensor interface data program.
- e) A Phase II Test Plan for a full load endurance test for the combined operational and Sensor Data programs.
- f) A Phase II Certification Procedure for the full load endurance test of these combined programs.

Test documents were required to satisfy the NAVMAT Digital Computer Program Documentation Standards and NAVSHIPS 0967-011-0011 Documentation Standards.

4.2 Top-Down Concept Program Testing

Top-down software development required that the program design, production (coding and checkout), and formal testing be done in a top-down manner. The principle advantages of top down development were:

- a) The ability to establish multiple fixed and visible milestones within the production and testing of a computer program.
- b) Early production and verification of the control logic which enabled a program to execute as a component of a data processing system.
- c) The ability to begin system integration and testing prior to the completion of program production.

Following program design, program production and testing proceeded element by element downward from the top. After the proper operation of each element was verified, that element was integrated into the software system which, when completed, became the deliverable program. During the software system development, the program elements which had not yet been produced and integrated into the system were represented by stubs. In the object program the stubs satisfied the element calling (control passing) functions depicted in the tiered hierarchy. A stub may also have modeled the consumption of computer resources, the occupation of main memory, and processor time use. Throughout program production, the combination of completed elements and stubs provided a complete representation or target machine model of the operating program.

To define a "level" for a given program module, a set of elements was selected for production. The selection was based on objectives of program operation and testing as assigned to the level.

4.2 (continued)

Some deliverable levels of the programs provided complete implementation of selected functions. Normally, the elements selected for completion at a level were selected from the top of the design hierarchy down, such that any planned-to-be-completed element had only completed elements above it. At each completed level, the existing coding and the remaining stubs provided a target machine model of the complete program. For example, when the modules of a program were completed to delivery at Level 1, they were integrated into a system and executed with the executive, a simulation program, and data extraction software. The ability of each module entrance to properly cycle through both the completed and the stubbed design objects was tested. Those functions which were complete at this level were evaluated, and the control logic and connectivity of the entire program was evaluated. When the testing of this integrated set of Level 1 modules was completed, the resulting Level 1 program was delivered. Testing of subsequent levels was facilitated because each testing phase was based on the previously validated level. Beyond Level 1 the tests themselves were built upon the tests performed at the previous level. Since each program performance requirement assigned and specifically documented each unique program element, faults detected by SQA tests were normally traceable to one specific segment of code. This gave a coherence to detected problems which minimized the time required for isolation and correction. Needed corrections and changes to design were most often made to a single element. This localization of affected code facilitated modifications, and helped to avoid partial (erroneously incomplete) corrections. The final level of testing was a complete functional test of the program's performance similar to that which would have been performed if the program had

4.2 (continued)

been performed if the program had been produced in the conventional manner. Each program level was deliverable following completion of testing. The individual deliveries were intended to facilitate system integration far in advance of the final completion of any of the many system programs.

Top-down design, production, and testing was employed for the Program 3 and 4 Projects.

4.2.1 Program 3

Program 3 consisted of:

- a) The Navy configuration controlled baseline Common Program operating system (CP) whose root was taken from Program 1, modified by ECPs and expanded to become Command and Control Operating System for a large system.
- b) Test software for this operating system.
- c) Simulation programs for intermodule interfaces, computer peripherals and the operational environment, (i.e., wrap around simulation and open and closed loop control).

4.2.1.1 Testing Objectives

Qualification testing for Program 3 required that test tools, test plans and procedures and simulation programs be generated for thorough testing of the developed software. A discipline was required at all levels of program development to ensure a reliable product.

The testing objectives ensured that the Command and Control Operating System satisfied the reliability requirements of real-time programs, met the requirements stated in the program performance and design specifications, and that they were structured in a manner to simplify life cycle maintenance.

4.2.1.1 (continued)

The programs were designed, developed, and tested in a top-down manner. This required that testing be performed at several "levels" of delivery. Breaking testing requirements for several "levels" facilitated testing of the control interface requirements early in the development cycle and allowed system integration testing to begin long before completely coded program modules of the software package were available.

To satisfy these objectives, the test documents described in paragraph 4.2.1.8 were generated. These test plans and procedures were designed to have a one-to-one correspondence between the individual test sections, and the customer approved performance requirements as stated in the computer program performance and design specifications.

4.2.1.2 Responsibility

Sperry Univac was responsible for providing the following test items for Program 3:

- a) Both informal and formal certification test plans and procedures for four levels of testing.
- b) Design, development, testing and documentation of a simulation program to provide a realistic environment in which to operate and test the operational C&CS.
- c) Support for customer testing and integration.
- d) Performance of verification and certification testing and generation of test reports.

The Common Program was a Navy furnished item and already certified program.

4.2.1.2 (continued)

The customer was responsible for review and approval of all design and documentation, and observed the correct operation of the formal qualification (acceptance) tests. The software was accepted when it operated correctly according to the qualification tests which conclusively demonstrated satisfaction of the requirements listed in the customer approved performance specification.

Program 3 documentation and testing satisfied the requirements of specification NAVORD Weapons Specification WS 8506, Revision 1, 1 Nov. 1971.

4.2.1.3 Scope and Extent

It was verified by Sperry Univac and certified by the customer that all code was executed under conditions to assure a documented degree of acceptability. These tests satisfied the requirements of the program performance and design specifications and demonstrated that the C&CS real-time operating system software functioned satisfactorily.

Specifications stated the testing requirements, method of testing, and the support programs and facilities necessary to ensure program quality. Program 3 used top-down design and development described earlier in this report and therefore, testing was required at four levels of development. These were:

- 1) Level 1 program testing prior to Level 1 Program delivery to ensure that the CPU, core, and mass memory utilization was as specified, and the I/O design was sound.
- 2) Level 2 Program Testing prior to Level 2 Program delivery to ensure that the primary algorithms of the program were operational, and the intermodule structure of the program modules was complete and operational.

AD-A040 049

SPERRY UNIVAC ST PAUL MINN DEFENSE SYSTEMS DIV
MODERN PROGRAMMING PRACTICES STUDY REPORT.(U)

F/G 9/2

APR 77 W E BRANNING, D M WILLSON

F30602-76-C-0136

UNCLASSIFIED

PX-11932-F

RADC-TR-77-106

NL

3 OF 5
AD
A040049





4.2.1.3 (continued)

- 3) Preliminary Program Testing (Level 3) prior to preliminary delivery of the entire package ensured that all of the functional requirements as stated in the program performance specification had been incorporated.
- 4) Qualification testing (Level 4) formally demonstrated that the program fulfilled the performance requirements and was the means by which the program was formally accepted by the customer.

4.2.1.4 Levels of Testing

Program modules of Program 3 were tested both as independent units and as elements of the overall integrated operating system. A high level functional test was applied to the program taken as a whole and individual high level functional tests were applied separately to each subprogram unit (program module). In addition a low level functional test was applied separately to each function provided by a program module.

Sperry Univac was not responsible for the C&CS operational program integration but did provide support, documentation, and simulation tools to aid in the effort.

4.2.1.5 Test System Viability

Program 3 test software was based upon the test support software used in Program 1 and earlier projects, but Program 3 test software included tools for supporting the three levels of program testing such as test control, data recording, and data reduction software and for simulation programs to provide the data generation required at all levels.

The test system featured modular construction and was scenario controlled. These features made it amendable to many different

4.2.1.5 (continued)

hardware configurations, and expandable to simulate additional external equipments or internal software modules. The scenario control program utilized card deck input. This input form provided test flexibility and was easy to control. The information on each card represented an action which could have been input from the control console, but using a card input prevented inadvertent operator actions and provided repeatability and integrity to the controlled time sequence of input control data. In addition, the test and simulation software used the same basic operating system software as the system being tested. This simplified the operations of transferring simulation programs between intra-computer and ultra-computer simulation environments and eliminated the need of providing a separate operating system for the test software.

4.2.1.6 Design for Ease of Testing

Design of the Program 3 testing system was keyed to the ability to test multiple hardware and software configurations with ease of modification of the test system. Use of simulation and re-configuration simplified isolation of deficiencies in the faulty module(s). Card deck test scenario input provided a reproducible, easily changed method of test control.

Test results were extracted from the on-line system by a real-time data extraction support tool, written to magnetic tape, and printed on the on-line printer for quick determination of pass/fail.

A communication link provided centrally controlled access for test stimuli to the integral operation of the program under test.

4.2.1.6 (continued)

Elements of the simulation program could be run simultaneously in the same computer, as the program to be tested, or in an independent computer. This provided simulation of all external and internal interfaces, functions and equipment.

The same operating system or a copy of it was in a different computer, was used to control both simulation and the program being tested.

In general the design of the Program 3 Simulation Program made it possible to "load" the C&CS operational program heavily with data from "live" or simulated sources. Actions could be simulated and tested on the C&CS operational program. The inter-module and intercomputer/peripheral message traffic could be traced by requesting use of the data extraction/reduction tools, and any other desired data could be obtained by using the debug aids, i.e., execution timing, dump, trap and snap capabilities of the Common Program.

4.2.1.7 Methods and Tools

The approved Computer Program Performance Specification (CPPS) was written by the Sperry Univac Program 3 development group on the basis of data furnished by the customer. The program requirements specified were reviewed and a section of the CPPS was written, giving testing and Quality Assurance (QA) requirements. This section formed the basis for the forthcoming test plans and procedures for each of the four levels of testing performed for Program 3. These were written by the qualification test group in Sperry Univac. The test plans for the ship-board release of the program were written by the customer.

4.2.1.7 (continued)

The test scenarios were created on punch cards and maintained as card decks as images on magnetic tape and as images on magnetic disk based upon the operational program documentation (CPPS, CPDS, User's Manual and test procedure). These test scenarios were input into the single or multiple computer configurations defined in the program testing along with data from real or simulated interfaces. In this manner the C&CS program was tested to the detail required to satisfy the CPPS and Test Procedure requirements. Test results were extracted using a data extraction module and debug aid capabilities. Extracted information was collected on magnetic tape, reduced to the output format desired, and written to a system printer for easy availability and analysis. The test results were documented in test reports with deficiencies documented as software problems, corrected by the development group and retested by the test group. This process was repeated until all problems were conclusively resolved.

Before release of the Shipboard issue of Program 3, concurrency tests and Factory Acceptance Tests (FAT) were required. The concurrency tests proved that two C&CS operational programs could operate concurrently in a multi-computer interface. The FAT test was written by Sperry Univac, although system integration and qualification tests were performed by the customer.

During Program 3 development, the Navy configuration controlled Common Program was modified via ECP's control. Testing for Program 3 involved testing the combined operating system and C&CS operational program, as well as the simulation and test software.

4.2.1.7 (continued)

Program 3 modules and systems were debugged at a government facility. Since time available was limited, a program checkout plan, checkout procedure, and checkout log were required to be filled out by each programmer using the test site before gaining access to the equipment. This allowed meaningful debugging to be performed using half hour or less time limits per programmer. Printed forms were used, but entries were hand written. See figures 4-1 through 4-3.

4.2.1.8 Testing Documentation

The following test documents were developed by Sperry Univac for Program 3:

- a) Computer Program Test Plans for both the Common Program and Program 3 for four levels of testing.
- b) Test Procedures for both programs for all four levels.
- c) Test Result Reports for testing of all levels.
- d) The Quality Assurance sections of the CPPS and CPDS.

Other documents were generated for the shipboard release of Program 3.

Test documentation was required to satisfy NAVORD weapons specification, WS-8506, Revision 2, 1 Nov. 1971.

4.2.2 Program 4

Program 4 consisted of a control application program as described in the Program Environment section of this document. This program executed under the control of the C&CS operational program developed in Program 3 and also used the Program 3 simulation resources. Testing for Program 4 was carried on at four distinct levels of delivery required to satisfy the systems top-down schedule design and development as described in the earlier section of this report and similar to that described for Program 3.

Checkout Item			Signoff			
No.	Description	T Y P E	Checkout Procedure Reference	Remarks	Programmer	Final Review
7.3.4.1	Enter JT Bias at XC Console (Button 07)	F		Bias in Ned. Trap Receipt of Message Type 47 at NR Module	<i>[Signature]</i> 1-11-72	
7.3.4.2	Console Ned to Binary Conversion Routine (INRCNED)	D	LP-21	Check Real-Time with Console for Final	<i>[Signature]</i> 1-11-72 MILLER 2-1-72	
7.3.5	Request JT Display at XC Console (Button 10)	F				
7.3.5.1	JT Display Updating (Button 11)	F		Observe Auto Updating Every 6 sec.		
7.3.5.2	Delete JT Display (Button 10)	F				

Figure 4-1. Typical Checkout Plan Sheet

BEST AVAILABLE COPY

Checkout Plan Item No. 7.3.4.2		Sheet No. LP-21 Prepared LKP 11-7-71	
Description Console NED to Binary Conversion Routine (NRCNED)			
Configuration 1. Manual c/o - NCRNED UPAK 2. Realtime c/o - NR, DB Modules, XC console on line		Initiating Action 1. Set up inputs in NED BCD format in NRNEDX and enter NRCNED from UPAK. <div style="text-align: center; border: 1px solid black; margin: 10px auto; width: 150px;"> <div style="display: flex; justify-content: space-around; font-size: 0.8em;"> 1010.1012.118.74.38 </div> <div style="display: flex; justify-content: space-around; font-weight: bold; font-size: 0.8em;"> NED4NED3NED2NED1NED0 </div> </div> 2. Use ENTER JT Bias function at XC console in normal mode.	
Response 1. Output in NRNEDZ. 2. Observe Bias Presentation at XC Console.			
How Implemented NRCNED converts 5 packed 8-4-2-1 BCD digits to binary integer. Uses reverse shift, multiply by ten and accumulate algorithm.			
Comments <div style="font-family: cursive; font-size: 1.2em; margin-top: 20px;"> Checked 1-15-72 RMS Error in zero value - Corrected </div>			

Figure 4-2. Program Checkout Procedure

KEY PUNCH	DK	CARD				INS	LABEL	OPERATOR, OPERANDS, AND NOTES	KEY PUNCH
		0	1	2	3	0			
	N	0	0	1	3	1	0	N, R, C, N, E, D, 6	⇒ IF Δ Δ Δ NR NEDX Δ NOT Δ Δ THEN Δ GO TO Δ
	R								⇒ NRCNED7 Δ S Δ Δ Δ CHECK Δ FOR Δ ZERO Δ INPUT
	B	0	0	1	3	2	0		⇒ SET Δ NR NEDZ Δ TO Δ Δ S
	L	0	0	1	3	3	0		⇒ GO TO Δ NRCNED9 Δ S Δ Δ Δ EXIT
									⇒
									⇒

ADDRESS	INSTRUCTION	NOTES	DELETE CARDS	INSERT CARDS
3552	530600 005101	JP to Patch	None	13.10
5101	100300 100010	Get NR NEDX		13.20
5102	510200 003555	Zero go to NRCNED7		13.30
5103	230300 100012	Not Zero - Zero NR NEDZ		
5104	530600 003553	Go to NRCNED9		

TROUBLE DESCRIPTION - Conversion algorithm needs special check for zero input.	
--	--

PROGRAM <u>NR MODULE</u>	
COMPUTER	COMPILE DATE <u>1-4-72</u>
SUPERSEDES LOG ON	
CORRECTION ON	
PAPER TAPE <u>X</u>	DATE <u>1-15-72</u>
MAG TAPE <u>B9</u> BLK <u>3</u>	DATE <u>1-17-72</u>
CARDS KEYPUNCHED	INITIAL <u>AMS</u> DATE <u>1-22-72</u>
DECKS UPDATED	UPR <u>1-24-72</u>
DOCUMENT UPDATED	

LOG ON 47 ORIGINATOR AMS DATE 1-15-72 DECK ID NRBL PROCEDURE NRCNED

PAGE 1 of 1

Figure 4-3. Checkout Log Sheet

4.2.2.1 Testing Objectives

Quality Assurance provisions for Program 4 required that test plans and procedures be generated for thorough testing of the developed application software. The disciplines described in the section on Top-Down Development were applied to all levels of the program's development effort in order to ensure a reliable product.

The testing objectives ensured that the application program satisfied the reliability requirements of real-time programs, that the program performance and design requirements were met, and that the application and test programs were maintainable for life cycle maintenance and support of Program 4. Test Plans and Certification Procedures were written with a one-to-one correspondence between each section of tests and the approved performance requirements to ensure all were met.

A further objective of the test documents, was to present the tests clearly and objectively. Sufficient explanation was provided to assure a clear understanding by the personnel conducting the tests.

4.2.2.2 Responsibility

For Program 4 testing, Sperry Univac was responsible for:

- a) Simulation programs for level 1 testing.
- b) Test control software, data extraction and data reduction programs for all levels of testing.
- c) Computer Program Test Plans and Test Procedures for all levels of testing.
- d) Factory Acceptance Test Software and documentation for levels 1, 2 and 3.

4.2.2.2 (continued)

- 5) Testing and preparation of test reports for levels 1, 2 and 3.
- 6) Support of customer testing for level 4 and support of subsequent testing and integration.

The customer provided the real-time simulation programs used for testing after level 1, wrote the Factory Acceptance Test, and conducted testing for final acceptance and integration.

Program 4 software development and Program 4 test software development were the responsibilities of two separate organizations at Sperry Univac.

4.2.2.3 Scope and Extent

The testing performed for Program 4 included formal qualification testing by Sperry Univac and formal factory acceptance testing by the customer. The formal qualification tests executed all code under conditions to assure the specified acceptability. These tests satisfied the requirements of the Computer Program Performance Specifications (CPPS) Quality Assurance (QA) provisions.

The QA provisions stated the testing requirements, method of testing, and defined the support programs and facilities required to ensure program quality. Program 4 used top-down design and development, and therefore, testing was performed at four levels of development as follows:

- 1) Level 1 program testing prior to delivery of Program 4 level 1 to verify the actual coding of all entrances and to exercise all resource allocations of the application software. This was to ensure that the core and mass memory utilization was as specified and that the I/O design was sound.

4.2.2.3 (continued)

- 2) Level 2 program testing prior to Program 4 level 2 delivery ensured that the primary algorithms of the program were operational and the intermodule structure of the program modules was complete and operational.
- 3) Program testing prior to Program 4 level 3 preliminary delivery of the entire program package, ensured that all of the functional requirements had been incorporated as stated in the CPPS.
- 4) Program Qualification testing formally demonstrated that Program 4, in a simulated environment, fulfilled the performance requirements specified in the CPPS and was used for program acceptance.

These levels were tested while under the control of the Program 3 operating system and used simulation and various test support tools as described in this report (see section 9).

Documentation (test plan, test procedure, and test reports) was prepared for each level describing the testing and identifying the functions tested.

4.2.2.4 Levels of Testing

Program 4 was tested both as an independent unit and as an element of the overall system. A high level functional test was applied to the overall program taken as a unit. In addition, a low level functional test was applied separately to each function provided by a program module.

Sperry Univac was not responsible for program integration into the live system, but did provide support, documentation, and simulation tools to aid in this effort.

4.2.2.5 Test System Viability

Program 4 test software extended and modified the test support software used in Program 3 and earlier projects. This included tools such as test control, data recording and data reduction to support the testing activities, and extended simulation capabilities to provide the data generation required at all levels. Since the test software operated under the same executive as the application program, and was written in the same language, no specialized peripherals, computers, compilers or executives were needed.

The test system featured modular construction with independent subsystems and was scenario controlled. These features made it amenable to many different hardware configurations and easily expandable to simulate additional external equipments and internal software modules, or to integrate with other test systems. The scenario test control system utilized a card deck input which simplified modifications, was easy to control and not susceptible to inadvertent operator actions as a scenario would be if input from a display console. The test and simulation software used the same operating system executive as the application program being tested which simplified development of the test system.

4.2.2.6 Design for Ease of Testing

Design of the Program 4 testing system was keyed to the ability to test multiple hardware and software configurations with ease of modification of the test system. Use of simulation and re-configuration simplified isolation of deficiencies to the faulty module(s). Card deck test scenario input provided a reproducible, easily changed method of testing and test control.

4.2.2.6 (continued)

Test results were extracted from the outline system by a real-time data extraction support tool, written to magnetic tape, and printed on the on-line printer for quick determination of Pass/Fail.

A communication link provided centrally controlled access for test stimuli into the internal operations of the program under test.

Elements of the simulation program could be run simultaneously in the same computer as the program being tested, or in an independent computer. This transportability of software allowed simulation of all external and internal interfaces, functions and equipment and allowed easy switching via recabling from simulated to live equipment for easier integration. The same Program 3 operating system software or a copy of it if in a different computer, could be used to control both the simulation program and the program being tested.

The design of the Program 4 test control and simulation software allowed the operator to "load" the application program as heavily as desired with test data from either "live" or simulated sources. Output of test results was obtained via the on-line real-time data recording module or by using the debug aids (dump, time, trap and snap) capabilities. Large amounts of data could be recorded in real-time on magnetic tape and dumped to the system printer off-line in a pre-established format.

4.2.2.7 Methods and Tools

The Navy approved Computer Program Performance Specification (CPPS) was written by the customer and contained a section on testing and quality assurance requirements. This section along with documentation standards per NAVORD Weapons Specification, WS-8506, Revision 1, 1 Nov. 1976, formed the basis for the forthcoming test plans and procedures. These were provided for each of the four levels of testing performed for Program 4. Test plans and procedures for the first three levels were informal. The formal Computer Program Test Plan (CPTPL) and Computer Program Test Procedure (CPTPR) were generated by Sperry Univac for the qualification tests of level 4. Design reviews were conducted by the customer against CPTPL and CPTPR. Sperry Univac conducted the tests for levels 1 through 3 while the customer performed the level 4 qualification test. Sperry Univac generated the test reports for all levels of testing.

The test scenarios were created on punched cards using high level commands as described in the simulation and test program operators and users manual for Program 4. The scenario "automated test" decks were debugged, and then served as input along with simulated and real data to test the application program. Using a data recording program, test results were either printed on-line, or recorded on magnetic tape and then reduced off-line to the desired format and printed. The results were analyzed, errors detected, STR's generated, STR's resolved, and the tests rerun. This process was repeated until all tests passed. The customer then observed the testing, reviewed and approved the test results, and closed the generated Software Trouble Reports (STRs). In this manner the Program 4 application program was tested to the detail required to satisfy the CPPS and test procedure requirements.

4.2.2.7 (continued)

The results of these tests were documented in test reports and any deficiencies corrected and retested. Annotated scenario listings were included in the test procedures. As an aid in analysis, test results were time-tagged and given mnemonic descriptors that were explanatory of the test function being demonstrated.

In general, the testing philosophy was to automate scenario driven tests to limit operator intervention as much as possible.

Program checkout plans, checkout procedures and checkout logs were informal. See figure 4-4 for a portrayal of simulation and testing functional interfaces.

4.2.2.8 Testing Documentation

The following test documents were developed by Sperry Univac for Program 4.

- 1) Computer Program Test Plans and Test Procedures, informal for the first three levels and formal for the qualification test.
- 2) Test result reports for all levels of testing.
- 3) Simulation and test program operators and users manual. (This provided a high level capability to generate scenarios.)

Test documentation was required to satisfy NAVORD Weapons Specification WS-8506, Revision 1, 1 Nov. 1971, documentation standards.

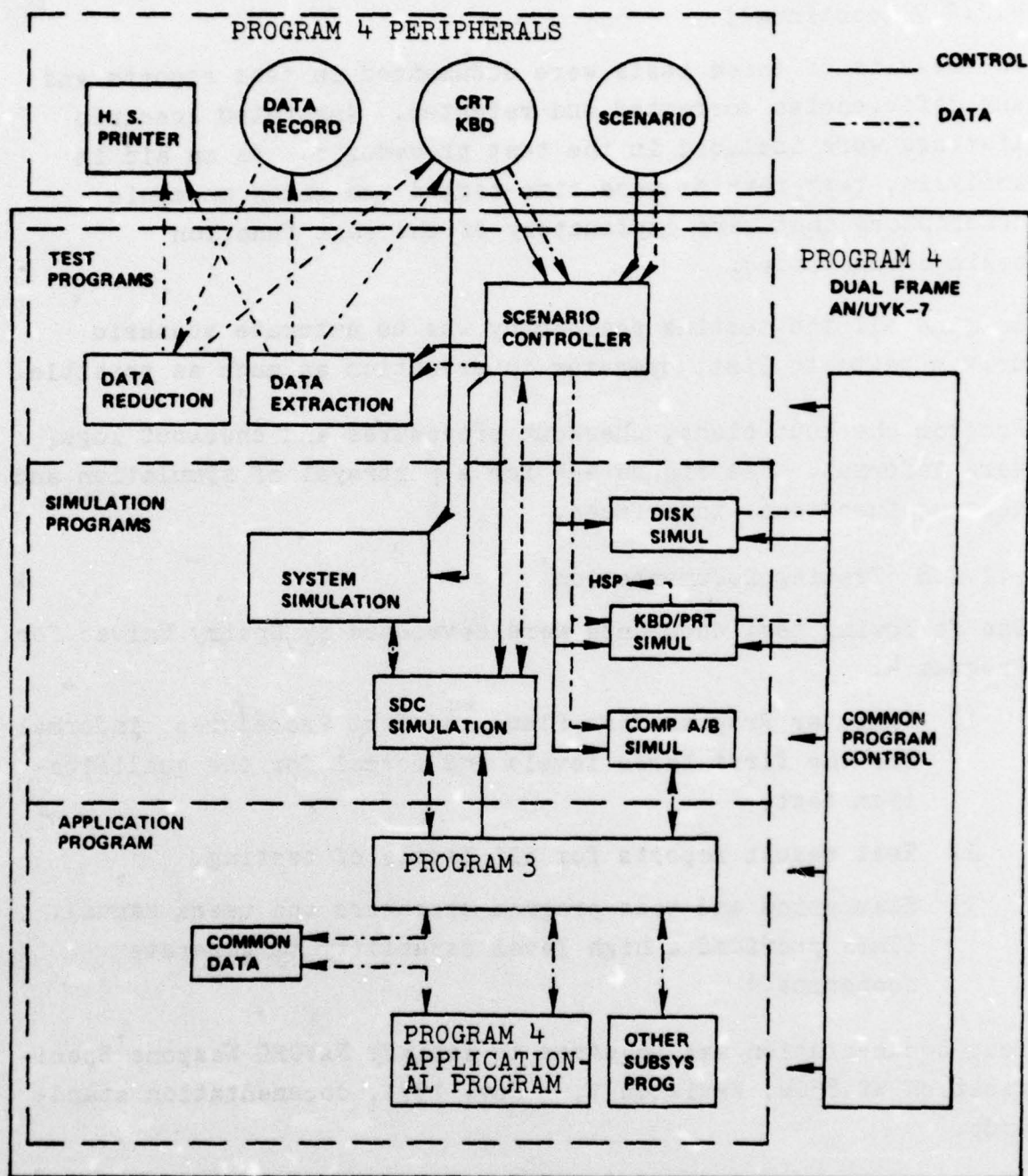


Figure 4-4. Simulation and Testing Functional Interfaces

4.3 CONCLUSIONS AND RECOMMENDATIONS

The purpose of program testing, regardless of the method used, was to locate and correct errors in the design and coding of the software system, and to complete a contracted milestone for software acceptance. Testing was successful on each of the four programs studied. The relative merits and conclusions reached are listed as follows:

- 1) Principle advantages of top-down design and development were:
 - a) The ability to establish multiple fixed and visible milestones within the production and testing of a computer program.
 - b) Early production and verification of the control logic which enabled a program to execute as a component of a data processing system.
 - c) The ability to begin system integration prior to the completion of program development.
 - d) A tendency to localize related functions, rather than scattering them through large portions of a program. This usually allowed function testing at a module level.
 - e) The ability to test completed program modules as a whole using stubs independent of the production of lower level elements.
 - f) The knowledge when testing an element that all higher level modules have been tested.
 - g) The top-down testing concept could be used even if conventional design is employed, however this loses many of the major benefits of the top-down design concept.

4.3 (continued)

- h) The ability to generate a test capability for level 1 and to expand and use it for more complex levels.
- 2) Similar to traditional design, top-down design is susceptible to changes in system requirements. Some re-coding may be necessary when early assumptions did not answer the final requirements.
- 3) Test control software driven by scenarios were used on all four programs to control automated testing. At the beginning of Program 1 development, scenarios were embedded in testing programs written uniquely for each program module and functionally tested. When it was realized that test system control with a scenario was actually a software tool, Sperry Univac created the current automated test control software and input scenarios providing for:
 - a) Test Automation.
 - b) Repeatability.
 - c) Test System Viability.

The earlier scenarios were punched on paper tape, however, due to the problems inherent with trying to generate and read reliably large reels of paper tape, later scenarios were generated on punched cards and usually maintained on magnetic tape, or magnetic disk file.

- 4) "Scripted" scenario test control with the test inputs consisting of timed operator key presses and other condition/response extractions when written to magnetic tape, formed a dynamic easily modifiable repeatable test tool. However, the operator had to exercise care and the control program had to contain safeguards against the entry of erroneous commands. Entry of scenario test

4.3 (continued)

systems from a deck of punched cards offered stricter control while still being modifiable.

- 5) A scenario generation high level language provided simplification of the scenarios.
- 6) Testing was both "open loop" (output subject to analysis only), and "closed loop" (output both available for analysis and available for dynamic feed back to effect the next test).
- 7) Simulation programs simulated not only intermodule and standard computer peripheral interfaces, but also employed "wrap-around simulation" to simulate the entire real environment enveloping the system.
- 8) A steady evolution was noted in the quality and capability of the test procedures. Additional tests were added, actions taken and observations of results were made clearer and more explicit; notes were added, expanded, and clarified to supply background information for each test.
- 9) Test support software (e.g., test control programs, data recording modules, data reduction modules, etc.) was designed for viability and became tools to facilitate their use on other projects and required only minor modifications or additions.
- 10) The test software was designed for testing in multiple configurations, using multiprocessors, multiple computers, and a wide variety of peripheral equipments. This gave the capability to reconfigure a system for faster check-out of system problems. The simulation and test software could be executed either in the same computer as the software under test or in an external computer cabled to the tested unit. This allowed the testing of

4.3 (continued)

the configurations prior to live integration into the operational environment.

- 11) The practice of "stubbing" undeveloped, but expected, functional "design objects" in top-down development allowed early testing of the system's responses to the expected timing and memory requirements.
- 12) The utilization of program checkout plans, checkout procedures, and checkout logs for program 3; allowed for up to twenty programmers to share four hour open shop time blocks effectively and to perform highly efficient program debugging. The advantages to be gained from using this requirement outweigh the difficulties attendant to changing personal attitudes and the costs involved with generating the forms and using them.

SECTION 5
SOFTWARE QUALITY ASSURANCE (TASK 2.4)

5.0 INTRODUCTION

Software Quality Assurance (SQA) is a discipline applying technical and administrative direction and control to assure that all software code and documentation delivered meets all specified and contractual requirements. These disciplines include the establishment of a SQA Program to cover each phase of software development (planning, analysis, design, production, test and configuration management).

Sperry Univac DSD management policies for product development include software quality assurance for all items delivered on a contract. Each division providing software development includes a functional organization chartered to provide SQA.

The following paragraphs described SQA as implemented by Sperry Univac DSD currently on Program 3. The SQA discipline was obtained by implementation of the following controls:

- . SQA Program Establishment
- . Design Control
- . Development Control
- . Testing/Certification Control
- . Program Records and Change Controls

The elements that comprise these SQA controls are described in terms of technical management objectives set for each element and procedures instituted to assure these objectives were met.

5.1 SOFTWARE QUALITY ASSURANCE PROGRAM

The SQA Program was established to specify organizational and functional accountability requirements necessary to implement software quality assurance.

5.1.1 Objectives

The overall objectives of the SQA Program were to:

- a) Promote the timely delivery of the contracted software and assure full compliance with all design and performance requirements.
- b) Make timely provisions for the tests, special controls, schedules, and documentation required to assure software quality.

5.1.2 Procedures

Planning the SQA Program involved:

- a) Identifying the elements of SQA in the formal SQA Plan for the project. This plan satisfied the MIL-Q-9858A quality assurance requirements and defined the program records and change controls to be exercised during the design, production, test phases and configuration management for the specific software development.
- b) Identifying the program records and change controls associated with deliverable software code and documentation (see paragraph 5.5).
- c) Establishing that the SQA group would provide further delineation, as required, to the contracted schedule of deliverable items.
- d) Establishing that the SQA Plan be implemented by the SQA organization chartered with the SQA responsibility.
- e) Providing technically competent personnel to perform the SQA functions with freedom to make decisions necessary to attain a quality product.
- f) Providing sufficient freedom of activity to ensure that quality requirements are consistently maintained.

5.1.2 (continued)

- g) Assigning defined responsibility and authority for SQA personnel to identify and evaluate quality problems, and to initiate, recommend, and provide solutions.
- h) Assisting in the establishment of the test requirements, identification, and specification of the testing to be performed, identifying test tools which may be required, and establishing the evaluation criteria for certification and acceptance of the deliverable software.
- i) Typical Sperry Univac DSD functional and project organizations for software development contracts are presented in figures 5-1 and 5-2 as established to provide technical engineering and management to the project.
- j) When more specific SQA procedures were required than those stated in the SQA Plan, the SQA group generated these as a part of Software Development Instructions, (See Software Development Standards section for a description of software development instructions.)

The resulting SQA Plan was reviewed and approved for implementation by the customer as satisfying the needs of the contract quality assured operational software.

5.2 DESIGN CONTROL

The SQA group worked independently of the project software development group during software analysis and design. This SQA work specified controls necessary to ensure program design and design documentation to satisfy the program mission. SQA conducted a critical review of the developed interface design specification and the program specifications to remedy discrepancies.

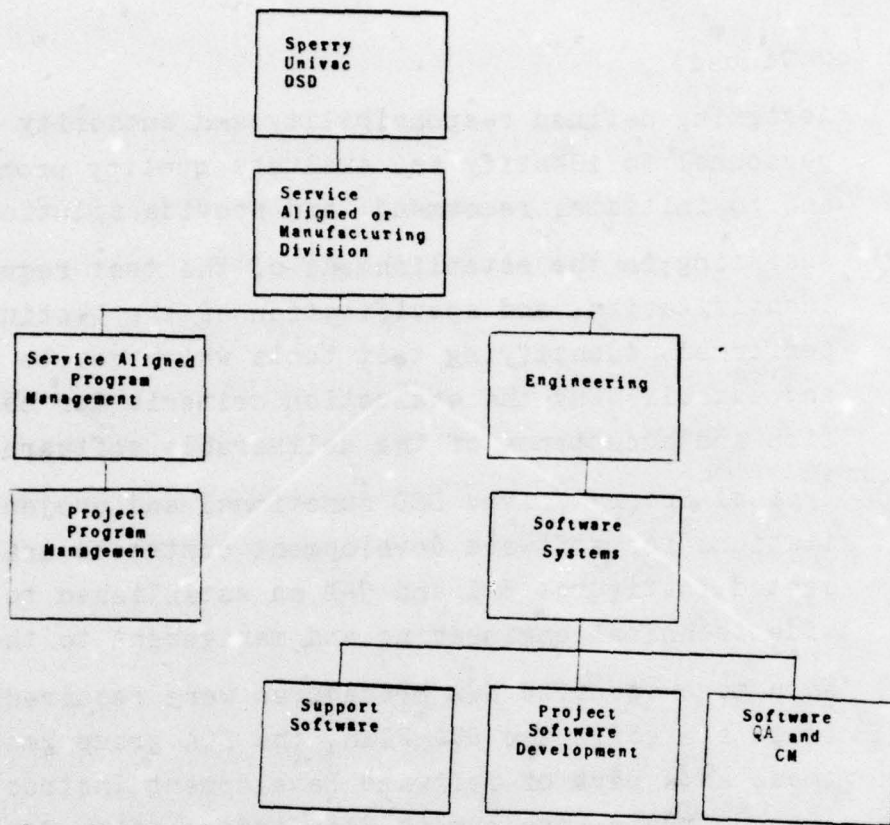


Figure 5-1. Typical Sperry Univac DSD Functional Organization

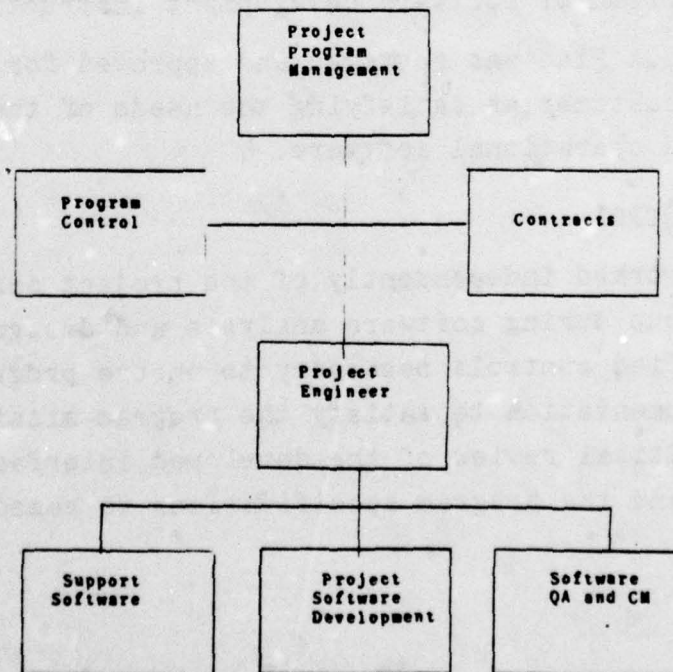


Figure 5-2. Typical Sperry Univac DSD Project Organization

5.2.1 Objectives

The objectives of SQA design control were:

- a) To assure quality program specifications for interfaces, performance and design.
- b) To assure quality and timeliness of program documentation.
- c) To assure functional adequacy (that the interface and performance specifications meet the supplied requirements, and the design meets these specifications) of contractually deliverable computer programs.
- d) To ensure that the documentation and design adhere to the established standards and conventions.
- e) To provide for identification and report the resolution of all specification and design discrepancies.

5.2.2 Procedures

The following procedures provided design control:

- a) The SQA group specified the requirements and criteria for program quality by providing material for the Quality Assurance Provisions section of the prepared program performance specifications, program design specifications and subprogram design documents.
- b) The SQA group identified the software requirements from the customer provided system specification requirements papers and provided assurance that the developed program interface design specifications and program performance specification met these requirements. A report was prepared documenting this critical review and was presented to the development group and the customer for inclusion in the Preliminary Design Review (PDR).
- c) The SQA group activity participated in design reviews, internal bi-weekly reviews, monthly customer reviews and Navy sponsored design reviews during the program development.

5.2.2 (cont)

- d) The SQA group audited the program design specifications to ensure adherence to the interface design specifications, performance specifications, project design standards and conventions and documentation standards (WS-8506, Rev. 1). The report from this audit was presented to the customer for inclusion in the design review. Any identified deficiencies were satisfied by the next revision of the program design specifications.
- e) The SQA group prepared the Computer Program Test Plan (CPTRL) based upon the customer requirements and the program performance specifications which conformed to WS-8506, Rev. 1 documentation standard. Upon review and approval by the customer and the Navy this plan was implemented. (See paragraph 5.4 for a detailed description of this implementation.)

5.3 DEVELOPMENT CONTROLS

The SQA group ensured that the program was developed in accordance with its governing functional and design specifications, and to ensure that a uniform method was established for controlling the object program during testing.

5.3.1 Objectives

The objectives of development control were:

- a) Identification of any operational baseline (configuration controlled starting point) source program.
- b) To provide in-progress inspections and controls to protect source program quality and functional integrity and to verify that execution timing and program sizing were within limits.

5.3 (continued)

- c) To provide proper changes to baseline module test programs and qualification test programs corresponding to changes specified for the operational baseline source programs.
- d) To ensure that proper coding procedures were followed as specified in the Project Standards and Conventions documents.
- e) To ensure that proper compiler and assembler techniques were followed as specified in the Project Standards and Conventions.
- f) To ensure that program debug/checkout procedures were used.
- g) To ensure that the program modules were produced by the approved top down process.

5.3.2 Procedures

The following procedures provided development control:

- a) The SQA group reviewed the Software Trouble Reports (STRs) against the baseline program and with the approval of the Software Change Control Board (SCCB) determined which program maintenance logs should be transferred into the controlled source. This effort provided a control and visibility to the customer on the resolution of STRs.
- b) The project engineer periodically inspected (minimum was bi-weekly) the listings of compiles to ensure compiles of the configuration controlled source were compiler error free, and did not exceed their core requirements (as set forth in the PPS). As the program was developed and approved Engineering Change Proposals (ECPs) were incorporated in the source code.

5.3.2 (continued)

- c) The SQA group maintained test records to show that each program module was coded and debugged, and had met the specified design requirements. However, SQA allowed the programmers freedom to initiate change request to:
 - 1) Improve the program's functional or operational design via ECP.
 - 2) Correct for logic errors.
 - 3) Correct for design deficiencies.
- d) The SQA group maintained records identifying updates, corrections, and compiles for changes made to programs during checkout and qualification testing that were within the approved design and had been authorized by programming supervisors.
- e) The SQA group maintained records of all updates, corrections, and compiles for program changes involving design changes authorized by the SCCB.
- f) The SQA group performed audits on coding to assure compliance to the Project Standards and Conventions document on the following:
 - 1) Compiler Language Conventions.
 - 2) Coding format.
 - 3) Card identification and sequence.
 - 4) Initial commentaries.
 - 5) Annotation comments.
- g) The SQA group inspected listings to ensure compliance to procedures specified in the Project Standards and Conventions document for the proper method of:

5.3.2 (continued)

- 1) Building a source library.
 - 2) Editing the source library.
 - 3) Compiling from the source library.
 - 4) Building a subsystem program object library.
 - 5) Editing a subsystem program object library.
- h) The SQA group assured that the proper procedures for building a system disk and system tape as specified in the Project Standards and Conventions were followed.
 - i) The SQA group reviewed software documentation of flow charts, figures and tables, nomenclature, and use of computer-oriented terms to ensure the requirements of WS-8506, Revision 1 were followed in the PDS and CSDD documents.
 - j) The SQA group reviewed software production to ensure that each updated program module was designed utilizing a top-down, structured approach.
 - k) The SQA group inspected and reviewed all program debugging checkout records such as debug and checkout sheets, program errata sheets, and records and results of tests performed to ascertain readiness to go into the qualification test phase.
 - l) Before any program modules were turned over to SQA for qualification testing, the Project Engineer determined if they were ready and signed them over to SQA.
 - m) All Program Maintenance Logs (PMLs), Software Trouble Reports (STRs), and Engineering Change Proposals (ECPs) that occurred during debugging were handled as specified in the SQA-established Software Development Instructions.

5.4 TESTING/CERTIFICATION CONTROLS

Software Quality Assurance was incorporated into the computer program software by conducting a series of informal and formal qualification tests. The purpose of the test effort was to certify and demonstrate that the deliverable computer programs successfully fulfilled the requirements set forth in the interface, performance and design specifications.

5.4.1 Objectives

The objectives of qualification testing and certification controls were:

- a) To determine as a result of agreement with the customer, the testing to be performed for each level of the developed program. (See section on Top Down Software Development for a definition of each level.)
- b) To prepare and update, as result of review by the customer and the Navy, the CPTPL.
- c) To develop and maintain during use the Computer Program Test Procedure (CPTPR).
- d) To define all qualification test parameters.
- e) To conduct qualification tests for each level of produced programs.
- f) To write formal test reports after completion of the qualification tests.
- g) To assist the customer in development, operation and writing of the test results for the Factory Acceptance Test (FAT).
- h) To assure that corrections made during FAT did not exceed a maximum of 25 program maintenance logs items and a maximum of 155 corrected cells (on approximately 25,000 words of object code).

5.4.2 Procedures

The procedures for qualification testing and certification controls included the following:

- a) Program qualification testing was performed on newly developed programs to the three degrees defined below. Program certification (repeat of qualification testing to baseline configuration controlled programs) was performed to the degree defined in the contracted work to fulfill an ECP.
 - 1) Module Unit Testing - Exercise the computer program on a module basis, determining that each component fulfilled its functional and design criteria. Each module was tested informally against the established baseline in its own test environment.
 - 2) Module System Testing - Exercise the computer program in a pre-defined series of steps in which a group of modules were tested together. The module was exercised through a function-by-function test while interacting with other new modules. As other modules completed Module Unit Testing they were added to the system. System Testing was repeated until the entire system was tested.

All tests were rigidly defined in the Computer Program Test Plan and performed in accordance with a set of test procedures based on the CPTPL at the Land-Based Evaluation Facility (LBEF). During Module System Testing, each module was demonstrated to meet its functional and design criteria.

5.4.2 (continued)

- 3) Factory Acceptance Testing - Demonstrated through the performance at the LBEF of a formal, rigidly defined series of tests defined by the FAT CPTPL and CPTPR, that the computer software satisfied its total functional and design criteria. Upon successful completion of the computer program FAT, the computer program was delivered to the customer.
 - b) The SQA group was responsible for reviewing the definition of the tests, and the implementation and performance of the tests comprising each degree of testing.
 - c) Prior to initiating testing of the computer program, the program developmental personnel informally tested the program utilizing Univac checkout plans and procedures (checkout testing).
 - d) Informal checkout testing was also performed on all support software exclusively developed for use with the deliverable computer program by the support software personnel.
 - e) The SQA group performed preliminary testing on support software prior to integration with the deliverable software. Support software was considered acceptable for integration when the support software was capable of performing the tasks required for the successful conductance of the computer program test procedures.
 - f) Support software was developed to aid all levels of testing and included the following programs:

5.4.2 (continued)

- 1) Interface Driver Programs - Special-purpose programs that determined the data transfer capability with respect to a specified computer-peripheral equipment interface or an intercomputer interface. These programs transferred known stimuli to drive the desired interface and compared the results that occurred to specific, predicted values.
- 2) System Simulation Programs - Dynamic, real-time programs, interfacing with the computer program, that provided simulated interfacing and functional data thereby simulating system hardware equipments on other system computer programs. Wrap-around simulators provided users with the capability to determine that the computer program satisfied its system functional and loading requirements in the absence of the system environment.
- 3) Test Support Software - Programs that were test tools, utilized to perform the following:
 - a) Extract test result data for analysis.
 - b) Reduce extracted test data.
 - c) Provide interfaces to specialized test equipment.
 - d) Prepare test scenarios based on operator-supplied input.
 - e) Automatically analyze the resultant test data for subsequent operator presentation.

5.4.2 (continued)

- g) The SQA group reviewed the support software requirements in the CPTPL and determined any additional detailed support/simulation software functional requirements simultaneously with the identification of additional computer program test requirements.
- h) The SQA group reviewed the development of the test plan for each computer program specified in the contract statement of work. The Computer Program Test Plan established the computer program qualification criteria for determining that the computer program met all the requirements of its functional and design specifications.
- i) Formal Computer Program Test Plans were prepared for the system and factory acceptance degress of testing.
- j) The SQA group developed a Computer Program Test Procedure for each computer program module for which a Computer Program Test Plan had been developed. The Computer Program Test Procedure provided the detailed step-by-step procedures and the information necessary for test personnel to evaluate the operation of the deliverable end-item computer program to ensure that the program had met the acceptance criteria as presented in the test plan. Successful performance of the qualification tests contained in this document demonstrated that the computer program satisfied its functional and design requirements as specified in the governing program documents (CPPS and CPTPL).
- k) The Computer Program Test Procedure was prepared in accordance with Navord documentation standard WS-8506, Rev. 1.

5.4.2 (continued)

- 1) The steps of the qualification test were listed in the order in which they were to be performed. The method to record the test results together with appropriate comments were included in the body of the Computer Program Test Procedures.
- m) The Computer Program Test Procedure contained as a minimum, the following characteristics that individually and corporately established the qualification resources to be utilized in determining the functional quality of the developed computer program:
 - 1) A list of equipments and the hardware initialization and shut-off procedures for each of the equipments.
 - 2) A list of programs, program load, and initialization procedures for each program.
 - 3) Personnel required to perform the testing.
 - 4) The test procedure itself, consisting of the detailed step-by-step procedures to implement the test scenario and to identify each necessary test input and the resultant action(s) that should occur to demonstrate that the certified tested program fulfilled its functional and design criteria.
 - 5) The data extraction/reduction requirements associated with the qualification test, and the analytical criteria to be utilized in determining the validity of the extracted data were presented and consisted of the following data:

5.4.2 (continued)

- a. Identification of the extracted data.
- b. The accuracy and frequency of the extracted data.
- c. The format of the reduced data.
- d. The criteria to be utilized in the analysis of the reduced data.
- 6) The SQA group supplied supportive documentation references utilized to produce the test procedures.
- 7) The SQA group supplied information required by the operator to distinguish between satisfactory and unsatisfactory results.
- n) The SQA group defined all test scenarios used during program qualification.
- o) Upon completion of the various levels of the computer program development effort, the computer program and all related test support and simulation programs for the particular level were presented to the SQA group for testing.
- p) The SQA group was responsible for test support and simulation software. They also determined the computer program development personnel required to support system testing.
- q) The SQA group assured that the computer program test procedures were correct.
- r) The SQA group determined the readiness of the computer program for system testing from the results of unit tests. Any discrepancies discovered were noted and the resultant problems were analyzed and were corrected during the initiation of system program tests.

5.4.2 (continued)

- s) Prior to formal demonstration of the developed operational program the CPP was placed under configuration control by the SQA group.
- t) The SQA group built a test system at the LBEF that provided the necessary hardware and software support requirements to conduct system testing.
- u) The SQA group conducted all system testing, logging all software troubles. All discrepancies in its operational and support software program were corrected, and the programs recompiled and retested as needed. Upon successful completion of system testing, the program became the baseline for factory acceptance testing.
- v) During the system test phase, SQA personnel performed the analysis of test results and defined the support necessary to complete the analysis phase.
- w) The SQA group was responsible for conducting, under the direction of the customer, program factory acceptance testing effort, and determining the support requirements necessary for the formal demonstration. Successful performance of the computer program during qualification demonstrated that the subject program fulfilled its functional and design requirements and that it was ready for delivery.
- x) During the factory acceptance test phase, SQA personnel performed the analysis of test results and defined the support necessary to complete the analysis phase.
- y) The SQA group prepared test reports after system and factory acceptance testing to document the results of the qualification effort. The test reports identified the computer programs tested and support software used, the computer program functions demonstrated, and the

5.4.2 (continued)

- y) result of the test of each computer program function.
The test report also identified any observed program design deficiencies against the design and recommended the necessary corrective action.
- z) The test reports recommend improvements which, if incorporated, would improve the developed computer programs or would enhance the software development process in future efforts.

5.5 PROGRAM RECORDS AND CHANGE CONTROLS

The following states the objectives and procedures that the SQA group utilized for the various program records and change controls maintained by SQA.

5.5.1 Objectives

- a) Program Test Records
 - 1) To provide objective evidence (test results) that the coded program complied with the program design document and that both complied with the requirements and criteria of the CPPS.
 - 2) To provide the data necessary for the evaluation and disposition of program problem reports found during testing.
- b) Program Software Trouble Reports
 - 1) To provide a record of all program problems encountered. (These reports were in-house records used to record problems during testing.)
- c) Engineering Change Proposals
 - 1) To communicate changes required to various program modules and documentation, which were over and above the CPPS and CSDD.

5.5.1 (continued)

- 2) To document the changes shown by Program Maintenance Logs which were the baseline source program.
- d) Program Maintenance Logs
 - 1) To ensure adequate recording of program problems with the corrective solutions.
- e) Audit Reports
 - 1) To provide a record of documentation and a record of discrepancies.

5.5.2 Procedures

- a) Program Test Records maintained for all tests included:
 - 1) Inputs (senarios and operator inputs).
 - 2) Annotated data reduction listings of results.
 - 3) Annotated teletype outputs.
 - 4) Test results (acceptance/rejection).
 - 5) Nature of changes and corrections made to the program and lists of the program errata.
 - 6) Test results of the program after incorporation of authorized changes (ECPs or program patches).
- b) SQA records were provided for review at the request of the customer.
- c) Program STRs generated during formal testing included the following:
 - 1) Site and data of occurrence or observation.
 - 2) Program module identification.
 - 3) Program compile/assembler or issue date.
 - 4) A concise but complete description of the problem.

5.5.2 (continued)

- 5) A statement of trial corrections or changes attempted and the result of each.
- 6) All Program Maintenance Logs associated with the problem.
- d) The SQA group monitored and verified the necessary corrective action and initiated program change requests before the Software Configuration Control Board (SCCB).
- e) ECPs were initiated to authorize the correction of assignable conditions adverse to the quality of a program. These changes included the following:
 - 1) Deletion/addition of a program item.
 - 2) Change sheets pertaining to program documentation.
 - 3) Computer program patch to be entered into the operating program and hardcopy listing.
 - 4) Availability date, test date.
- f) ECP records were maintained by the program library as part of the program package.
- g) An ECP report was not considered closed until all action was completed at all affected facilities.
- h) ECPs were approved only by the SCCB.
- i) The SQA group ensured that program changes occurring prior to program qualification were included in the Computer Program Test Procedures.
- j) The SQA group used Program Maintenance Logs to control the object code once the module was identified on the controlled program library.

5.6 TECHNICAL ORGANIZATION RESPONSIBILITY

Software Quality Assurance, Support Software Development, and Configuration Management (CM) Controls were implemented through a functional, distinct organization, separate from and independent of any related computer program development effort. SQA organization was compatible with, and subject to, all defined standardized management practices and guidelines established within Sperry Univac DSD/Engineering.

The areas of responsibility for each of the three groups (SQA, CM, and Software Development) concerned during the Operational Program development are shown in figure 5-3, Technical Areas of Responsibility.

All three groups were under the cognizance of the Program Manager and under the technical direction of the Project Engineer.

Following is a list of the major areas of responsibility of each group.

a) The Software Development group:

- 1) Designed, developed, and delivered the Operational Program.
- 2) Developed and delivered the Operational Program documentation (CPPS, PDS, CSDD, and Computer and Program Operators Manual, CPOM).
- 3) Provided technical support throughout the test phases.
- 4) Incorporated all program changes into the Operational Program.

5.6 (continued)

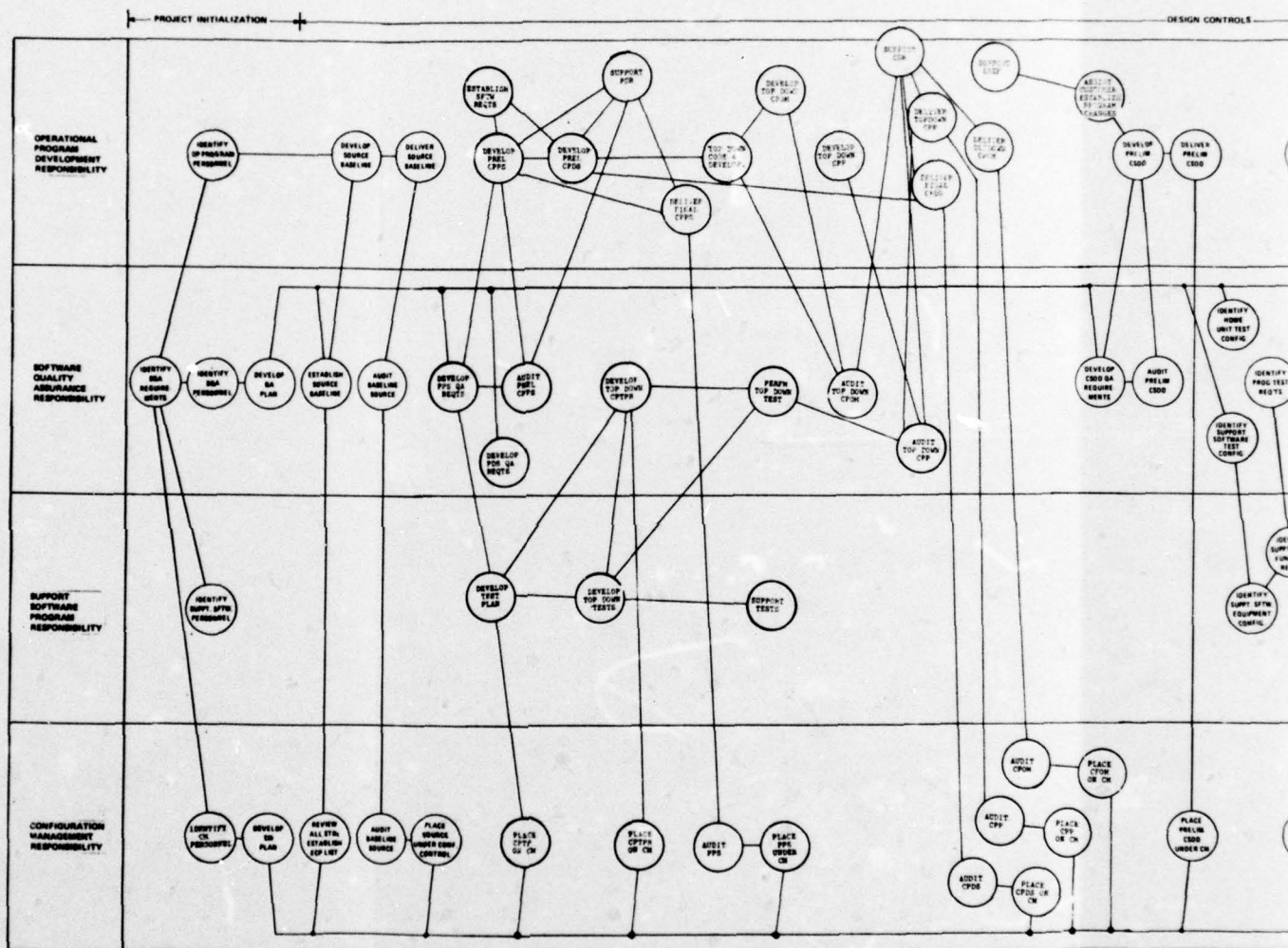
b) The SQA group:

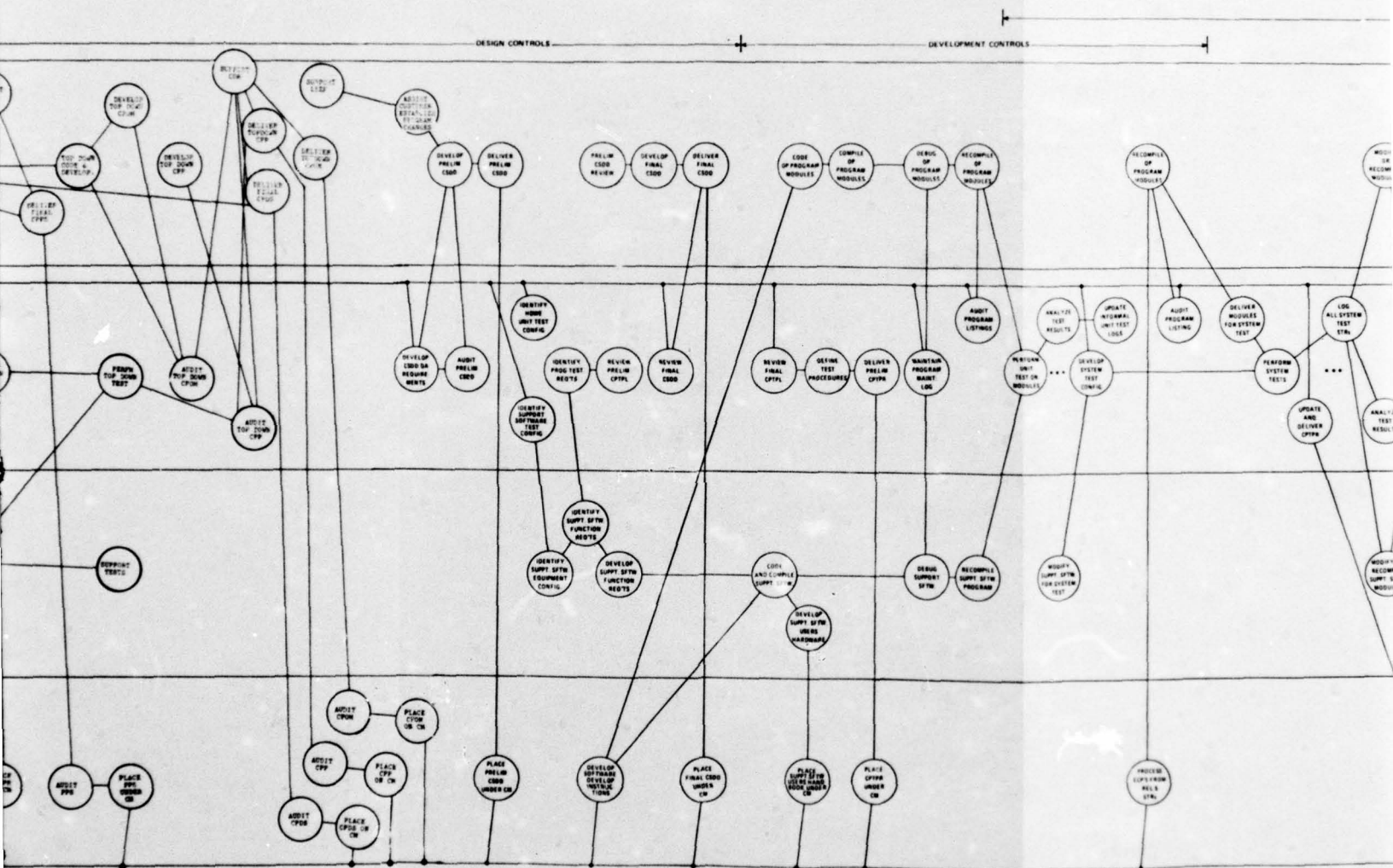
- 1) Produced, delivered, and carried out the Quality Assurance Plan.
- 2) Carried out the final review of all project deliverable documentation.
- 3) Reviewed the development of the Computer Program Test Plan.
- 4) Produced the Computer Program Test Procedures.
- 5) Conducted the testing and formal qualification of the developed Operational Program.
- 6) Recorded all test results and provided adequate testing and qualification reports.
- 7) Tested all changes made to the baseline program.
- 8) Provided schedules of all significant events if they were not delineated in the contracted schedule.
- 9) Produced and maintained records and reports.
- 10) Processed as defined in the Configuration Management Plan any STRs found during Module System Test.

c) The Software Configuration Management group:

- 1) Developed and carried out a Configuration Management Plan.
- 2) Placed all deliverable items under formal configuration control.
- 3) Processed and disseminated all ECPs.

BEST AVAILABLE COPY





2

BEST AVAILABLE COPY

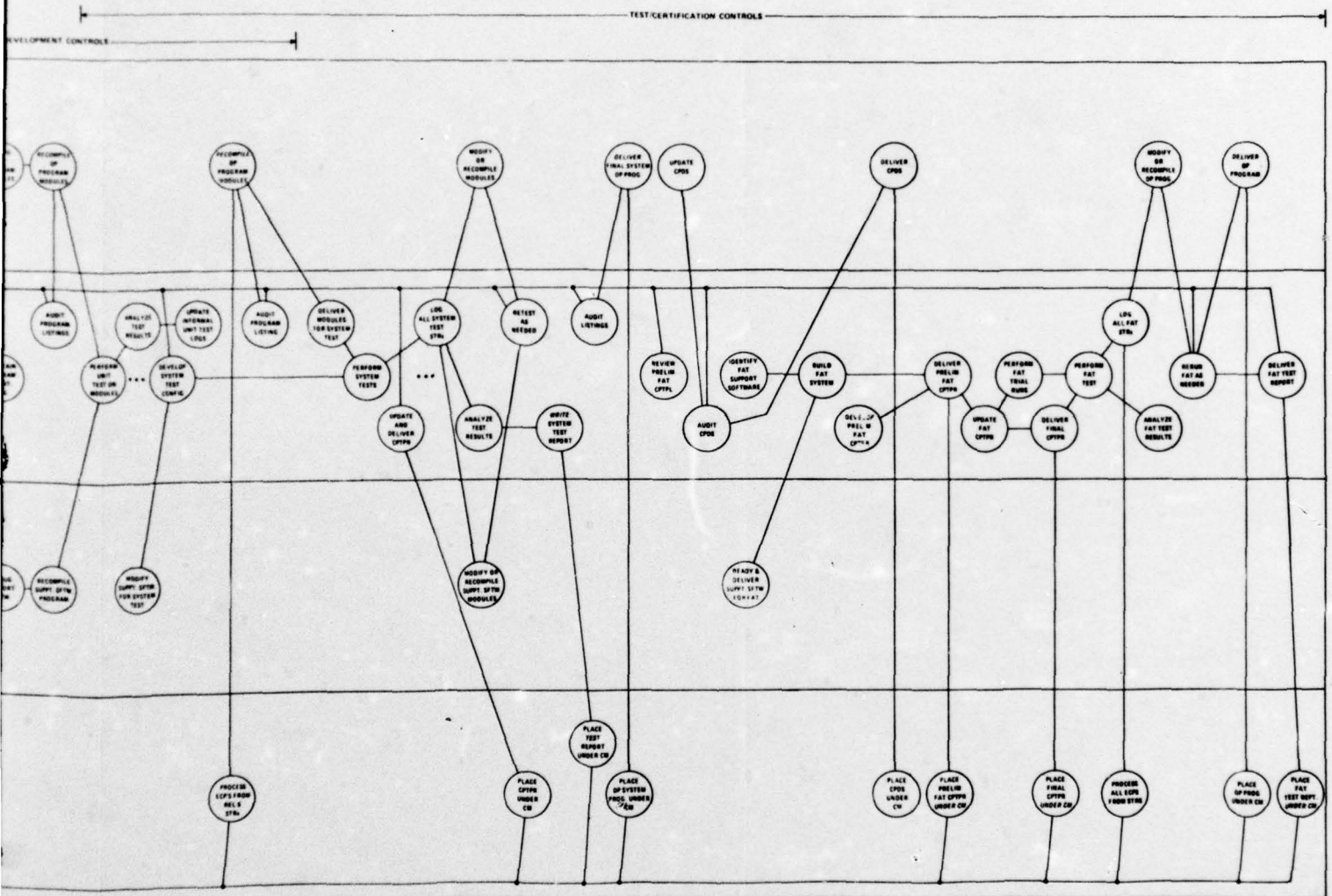


Figure 5-3. Technical Areas of Responsibility

5-23/5-24

3

5.6 (cont.)

d) The Support Software group:

- 1) Designed, developed, and maintained all supporting test software.
- 2) Provided technical support of test software throughout the test phases.
- 3) Developed and produced all support software documentation.

5.7 SUMMARY AND CONCLUSION

Sperry Univac's policy requires that Software Quality Assurance be proposed on all software developments. The initial implementation of software is a small part of the life cycle cost of software. SQA applied during development is a most important aspect of producing quality operational software. SQA further provides for enforceable configuration management practices well before final software delivery, and enables more easily maintained software over the life cycle.

Software produced under strict quality control as described on this example not only reduces the life cycle cost of the developed software, but allows for adaptation of the produced software to many related applications.

The SQA group proposed the recommended modifications and additions to the quality assurance program:

- a) SQA should have had a longer lead time to develop test support software. SQA was at times already running tests while support software was still being written and debugged.
- b) Since the SQA effort was largely a management of test operation at the LBEF, a resident Quality Assurance manager should have been provided by the SQA group.

5.7 (continued)

SQA concluded that the following tool, procedure, and method did increase efficiency, effectivity and viability on these projects:

- a) The scenario control program precisely timed the inputs for automatic control of test functions.
- b) The test plan precisely defined the degree to which software was to be tested.
- c) The top-down program development method succeeded as in section 2.

Sperry Univac software technology is evolving to increase the efficiency of software development by the use of general use systems and tools, to increase the software product quality by control throughout development, and to reduce the life cycle cost by producing manageable, well documented and viable software.

SECTION 6

SOFTWARE CONFIGURATION MANAGEMENT (Task 2.5)

6.0 INTRODUCTION

This section describes the Software Configuration Management (SCM) practices and procedures as employed by Sperry Univac in their development of Programs 1, 3, and 4 and the MODX2 Common Program Operating System. SCM was a separate contract item for each Program. The methods used and effectiveness of SCM on these programs is discussed to the extent of information available.

6.1 DESCRIPTION OF A REPRESENTATIVE SOFTWARE CONFIGURATION MANAGEMENT PROCESS

Software Configuration Management (SCM) was included in the Project Control System as a separate Work Breakdown Structure (WBS).

The programs under study used one of the Host Management Tools, CSTAR or PROMIS for this control (section 9). CSTAR, in particular, has received tri-service validation for compliance with DODI 7000.2 (AFSCP 173-5) requirements.

The SCM plan was the first milestone in the master schedule. SCM used the WBS as a tool to define and control the elements for development within cost and schedule.

SCM was in effect throughout the entire development period.

SCM is a discipline applying technical and administrative direction and surveillance to:

- a) Identify and document the functional and physical characteristics of a configuration item.
- b) Control changes to those characteristics.
- c) Provide a means for processing change requests, recording this change request processing, implementing approved changes, and reporting these changes in implementation status. For purposes of this report, these are referred to as the "Principle Objectives of Software Configuration Management (SCM)".

The Navy required that each contractor generate an SCM plan that satisfied these Navy's overall SCM requirements. The SCM plan implemented by Sperry Univac satisfied these requirements. A representative SCM plan used by Program 4 satisfied the following functions and objectives of the Navy's overall CM requirements:

6.1 (continued)

- a) System integrity was to be maintained by a systems-oriented review of all proposed changes which impacted the system specification or interface control documents.
- b) There was to be a centralized change control function in order to maintain orderly coordination, status, and scheduling of all proposed changes in the system.
- c) A screening of all desired changes before generation of a formal proposal was to be performed to filter out changes which were not likely to be approved.

The Program 4 SCM Plan provided rigorous control of software development accompanied by the mechanisms for upward transmittal of change requests. The software was labeled as a standard Computer Program Configuration Item (CPCI - see glossary for description) proposed subsystem changes were screened by the customer and processed via the procedure described in the SCM Plan. This assured that the software configuration management dovetailed with the program procedures and the overall Navy Configuration Management Plan.

Software changes were processed through a status accounting system which assured that the provisions of Navy SCM Plan were fulfilled with respect to each Configuration Item (CI - see glossary for description). In particular, status accounting was capable of:

- a) Providing a complete record of approved configuration documents for each CI within the subsystem.
- b) Recording all approved changes and field modifications by CI identification number and serial number.
- c) Providing the capability for verifying the incorporation of engineering changes to a CI on a serial number-by-serial number basis.

6.1 (continued)

d) Providing to the customer:

- 1) Status of Navy approved system level Engineering Change Proposals (ECPs).
- 2) Internal status of pending system level ECPs which were planned for submittal to the Navy for approval.

Preliminary and Critical SCM Plan design reviews were provided in accordance with the Navy and were described in the contract and SCM Plan. The reviews provided the customer with periodic monitoring of the status of the design and documentation of the software.

Status of the actual software development was monitored by means of the test regimen described in the Quality Assurance section; however, configuration management was applied to each software item as it was developed, according to the procedures described in the subsequent paragraphs.

6.1.1 Configuration Control of Computer Programs

Configuration control of code was an important, integral part of the total control scheme. Configuration control of code started when a programmer released the module or subprogram as debugged and tested and it was approved by the responsible manager. The card deck, source and object files for the approved module or subprogram were placed under control in the Computer Program Library (CPL). The official source of approved code was the CPL, and the code could be changed only through internal channels. Working or floor tapes could be changed as required. Because top-down design on Program 3 and 4 could be defined, developed, debugged, and tested individually, an optimum number

6.1.1 (continued)

for parallel production was selected; consequently computer program control could be generated early in its life cycle. All integrations and tests used only components under configuration control.

6.1.2 Computer Program Library (CPL)

The CPL functioned to provide acquisition, maintenance, and dissemination of baseline computer programs, data packages, associated documentation, and all technical data items. These services were provided during the development period and continued through the in-plant test during the LBEF and shipboard test periods. The CPL was custodian of all CPCI products placed under internal configuration management as a result of CPCI or document approval. The Program 4 regional data base was treated as a component and also placed under CPL control.

The functions to be performed by the CPL, CPL organization, and interaction with other organizational elements to which it provided by the CPL varied as the computer program development activity passed from definition stage to the development and test stages. CPL responsibilities during the two basic time periods -- prequalification and postqualification -- are identified in subsequent paragraphs.

Prequalification, the major activity of the CPL, supported computer program development by accumulating, controlling, and storing the software items used in the development and testing processing, including:

- a) All preliminary versions of CPCI computer program components, data, and associated documentation for which library storage and control was essential.

6.1.2 (continued)

- b) All supplied and internally developed utility and support programs and their associated documentation used in the computer program development process.

During the development process, the CPL provided change visibility for those computer programs and associated documentation which had been placed under internal control.

A master baseline copy of the CPCI product and documentation was retained by the CPL. These baseline copies, as modified by change pages, were used to support test activities and further modifications. The CPU provided record keeping functions associated with the processing of Computer Program Problem Reports (CPPRs) and Computer Program Change Orders (CPCOs). The relationship of the CPL to the ship support technical organizations involved in computer program activities and test operations involved in computer program activities and test operations is presented in subsequent paragraphs. During the development period, it was necessary to maintain copies of the program decks and tapes associated with preliminary versions of the CPCIs and support programs as they approached operational status. These copies were used to provide backup working duplicates to the program developers when required.

As the CPCI design approached completion, a point was reached where internal informal control of program decks, other than that by the Configuration Control Board (CCB), had to be established. This point was the time when a CPCI or component of a CPCI was debugged and individually tested and ready for the start of integration testing. Then, the developing organization transmitted the following to the CPL, along with a Release Notice:

6.1.2 (continued)

- a) A copy of the program deck (or tape) to be used in the test.
- b) A current computer program listing.
- c) Test procedures.

Whenever it was necessary to make a change to a given program deck, the following steps occurred:

- a) The development organization documented the problem on a CPPR and prepared a change package, including:
 - 1) A CPCO documenting the corrective action.
 - 2) Program corrector cards or revised decks.
 - 3) Listings.
 - 4) A test report covering test of changed program.
- b) The manager of the development organization signed the CPCO, indicating approval of the change and the change-package was delivered to the CPL.
- c) The CPL updated the index and program listing and document file by additions, deletions, or corrections.

These procedures assumed that at least one copy of a card deck (or tape), representing a program which has reached a significant development event, was always available at the CPL. This master deck was used solely for duplication purposes, and only the librarian was authorized to handle it or to provide change incorporation actions.

When in-plant verification had been accomplished for a given CPCI, the baseline master deck and/or tape, together with the computer program listing, were delivered to the CPL. The master copies were labeled and indexed by the CPL and placed in files. Master tapes or decks were never used for conducting tests,

6.1.2 (continued)

training, or operations.

The master copies of computer program products and documentation were retained in the CPL. The CPL distributed copies of the baseline master program decks to the locations where they were available to support the pertinent activities.

The CPL was the local point for receiving, logging and transmitting all CPPRs for review. After the customer initiated action on CPPRs resulting in program change actions (Specification Change Notices (SCNs) and CPCOs), the CPL arranged for updating master copies of decks, tapes, listings, etc., and transmitting them to all approved recipients.

6.1.3 CPPR/CPCO Processing

During the design, development, and testing of a computer program, changes to a controlled component or a CPCI were required. All problems encountered with controlled components or CPCIs were documented on a CPPR. All CPPRs and proposed CPCOs went through the CCB to ensure that the proposed change did not adversely affect other CPCIs within the program or interfaces with other computer programs or equipment. If a proposed change caused intermodule or subprogram effects, the CCB directed the cognizant programming groups to prepare CPCOs remedying any incompatibilities. The incorporation of changes into existing controlled tapes was done by the CPL, using patches or recompiles. If implementation of a CCB approved change required an ECP, the appropriate technical groups and the CPL were notified, and ECP and SCN action was started.

During the module/CPCI integration cycle, problems were encountered which required changes to the test plans, test procedures,

6.1.3 (continued)

and programs. Document changes were performed using the Technical Change Notice (TCN) and the Document Change Notice (DCN), and replacement pages in lieu of the SCN. A change to a component/CPCI was required to cycle through the entire change procedure, however, before the component/CPCI or program was officially changed.

Changes to components and CPCIs were unofficially made to a working or floor copy to expedite the solution to a problem; the change control procedure applied only to items under control of the CPL.

6.1.4 SCCE Processing

The Configuration Control Boards (CCBs) determined whether changes should or should not be authorized for modification of a program document or code baseline. A change could be initiated by a computer program contractor, a prime contractor, the procuring agency, or the ultimate user. Because of this change, control was maintained through the establishment of various levels of CCB's. The hierarchy had a chained relationship. Changes approved by the contractor CCB at the lowest level were submitted to the next higher level, the Prime Contractor CCB for approval, changes approved there were submitted to the appropriate Navy CCB board. This gave upward visibility to changes.

Each CCB consisted of representatives of the groups affected by the changes considered at that level.

6.1.4 (continued)

An ECP was classified as either a Class I or a Class II change. These classifications were based on the definitions contained in Military Standard MS-480 and Appendix XIII of MS-482. A Class I change satisfied one or more of the following conditions:

- a) Makes a significant change to the operational characteristics of the program.
- b) Adversely affects schedule or cost.
- c) Has a major impact on other system interfaces or equipment.

A Class II change is any change which did not fall within the definition of a Class I change.

6.1.5 Baselines

In general Navy controlled baselines were established as soon as a document or program satisfied certain requirements.

Performance Specifications, once approved through a PDR (Preliminary Design Review) became a "functional Baseline" and were under formal control from that time on. Design specifications approved at a CDR (Critical Design Review) became a "design baseline". The computer programs, as each was certified, became a "certification baseline". Upon acceptance of the integrated system by the Navy, all programs and documents were placed under control as an "operational baseline" while in actual use.

The basic sequence for processing change requests is shown in figure 6-1.

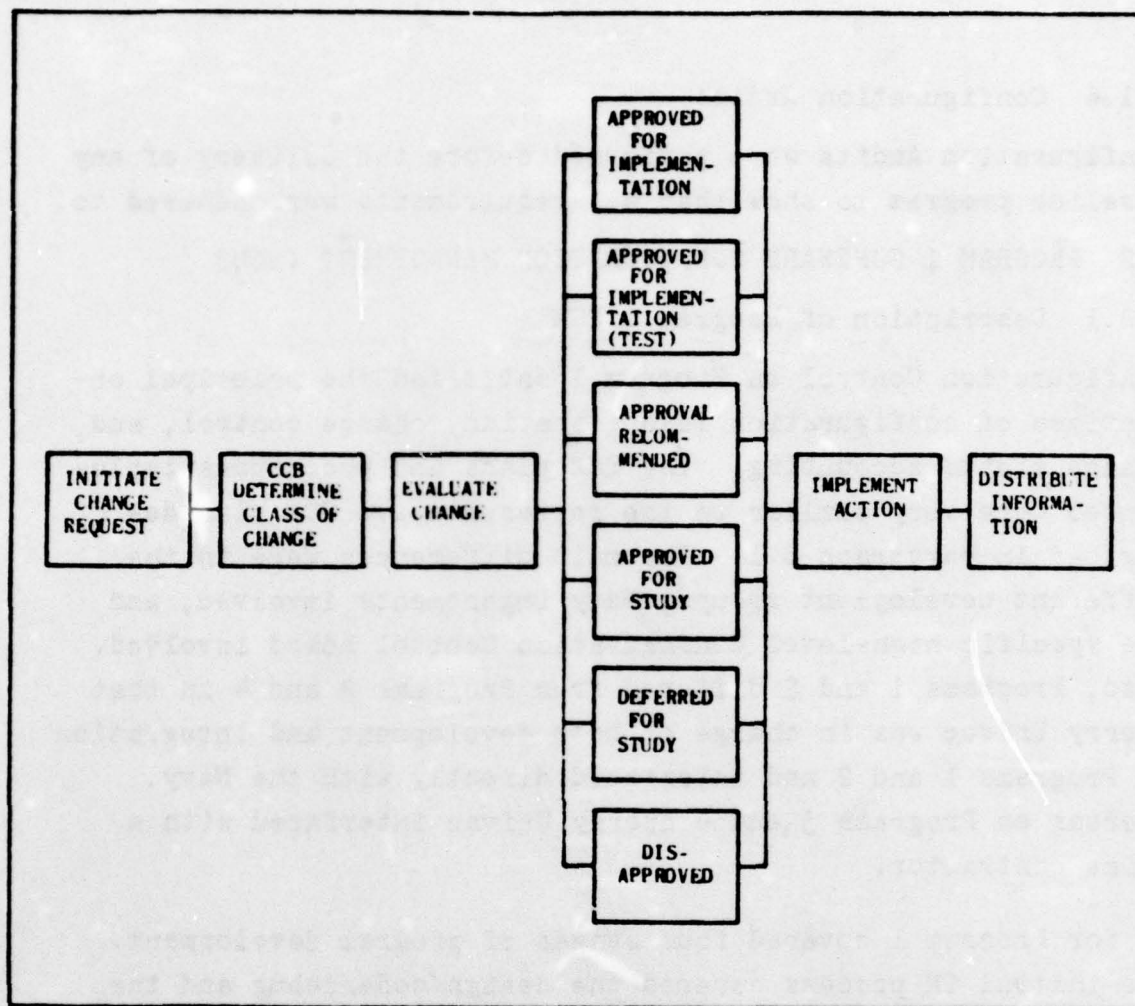


Figure 6-1. Representative Sequence For Processing Change Requests

6.1.6 Configuration Audits

Configuration Audits were performed before the delivery of any baseline program to show that all requirements were adhered to.

6.2 PROGRAM 1 SOFTWARE CONFIGURATION MANAGEMENT (SCM)

6.2.1 Description of Program 1 SCM

Configuration Control on Program 1 satisfied the principal objectives of configuration identification, change control, and change status accounting. The SCM plans and procedures implemented were very similar to the representative SCM plan described in paragraph 6.1. The main differences were in the different development groups, Navy Departments involved, and the specific high-level Configuration Control board involved. Also, Programs 1 and 2 differed from Programs 3 and 4 in that Sperry Univac was in charge of both development and integration on Programs 1 and 2 and interfaced directly with the Navy. Whereas on Programs 3 and 4 Sperry Univac interfaced with a prime contractor.

CM for Program 1 covered four stages of program development. The initial CM process covered the design/code/debug and the test/integration stages. This process was redefined and extended to provide what was called Interim Software Maintenance (ISM) CM during stage three (ship trials support). This stage lasted until the Program was fully operational in stage four. At that time Program 1 came under life cycle support maintenance CM control. The main differences between these CM processes were the changes from supporting program development to controlling program maintenance, and the different Navy departments which assumed responsibility for the different stages. Other differences were as follows:

- a) CM, during stages I and II, allowed either Class I or Class II changes. During the ISM period stage III, all changes are considered Class I and treated formally.

6.2.1 (continued)

- b) The SCCB for the original CM plan was responsible to a Configuration Management Committee for final approval or disapproval.

Program 1 did not recompile their baseline source code once system integration and evaluation testing commenced unless specific approved by the high level Navy SCCB. The source code, therefore, was frozen at that point so that any program maintenance consisted of patches to the subject code. The internal Sperry Univac SCCB used a CHECKREAD function available in the Utility Package (UPAK) to compare the current code to the baseline and list the differences. The listing consisted of all patches made since the last baseline was generated. A Program 1 Integration Operational Program Problem Log was maintained by the integration checkout programmers to record all discrepancies on a Program Problem Log form and all modifications (patches) to the current object code on a Program Maintenance Log form. The sum of the patches from the Maintenance Log must have matched the listing of patches obtained from the corresponding CHECKREAD listing.

Status reports were generated to record program patches, the area used, and patch area still available for use. The report referenced the Maintenance Log item associated with the patch and the program module to which it applied.

The status reports were generated and updated manually. They were reissued periodically for each of the major programs.

Configuration audits provided a formal examination of each Configuration Item. The audits performed are described in the following subparagraphs.

6.2.1.1 Physical Configuration Audits

A Physical Configuration Audit (PCA) was the means of establishing the current configuration identification of a computer program for test and acceptance as Configuration Item (CI). The PCA assured that the current configuration of the modules/segments of the CI program matched the module/segments of that programs baseline configuration identification, or that differences were reconciled.

PCAs were performed by conducting a CHECKREAD of the program's baseline against the then current program configuration, using the standard UPAK functions.

6.2.1.2 Program Verification Audits

The Software Integration Agent performed a program verification audit of the Program Baseline (PBL), Common and User programs, as each of these individually certified GFI programs were received. This audit was a two-way cross check verification of each baselined program. The advantage of this cross check verification was the assurance through demonstration that only approved program patches, and all lines of those approved patches were included, as recorded on the patch logs.

The program source record was assembled and a program tape was built of these newly assembled individual, Common and User programs. This newly built tape was identified as the Software Integration Agent's configuration controlled baseline assembly tape for that particular program. This baseline assembly tape was used for an initial PCA against the delivered Object Tape provided for that program. Listed differences between these program tapes were annotated with the identification numbers of the program patch logs delivered as part of the Program Confirmation Identification (PCI) of the program. Non-reconciled differences were reconciled by the responsible development agent.

6.2.1.2 (continued)

Additionally, a program product record tape, consisting of the baseline assembly tape plus PCI patch logs, was built by the Software Integration Agent. This tape was used for a PCA against the GFI Object Tape. Any listed differences were reconciled by the responsible development agent.

6.2.1.3 Informal Audits

The Software Integration Agent conducted periodic informal audits of the System Programs throughout the development and support of those programs. These audits were performed at either Navy or the development agents contractor's/subcontractor's GFE program development sites. The audits were performed as frequently as weekly or as infrequently as monthly, as the amount of program integration and test change activity fluctuated. In general, the more the change activity, the more monitoring for change and status accounting control.

The program product record tape, built by the Software Integration Agent, was used as the baseline for these audits. This tape was maintained and updated by the software integration agent, per SCCB control, to reflect program currency of the individual programs baselined configuration plus approved changes. All differences between the then current product record tape and working programs in use at that facility were reconciled jointly by the Software Integration Agent and responsible development agent.

These audits insured the distribution and inclusion of program changes at each operating site for the coordination of system program development and testing activities.

6.2.1.4 Formal Audits

The Software Integration Agent was responsible for conducting formal PCAs prior to the start and upon completion of formal certification/verification testing of the C&CS System Programs. As such, these audits were performed at sites. The various test and integration sites.

The formal PCA was performed by conducting a CHECKREAD of the baseline assembly tape against the subject certification/verification program, which provided a complete listing of program differences. These differences were annotated with the identification of the program patch and associated ECP numbers, e.g., (Log No.) / (ECP No.). All differences were reconciled jointly by the Software Integration Agent and the responsible development agent.

6.2.1.5 Audit Reports

An audit report, which documented the post certification/verification formal PCA and the reconciliation of differences in the CI program, was submitted with the test report for the program.

The SCCB minutes contained the results of the configuration audits and the determinations for unreconciled items.

6.2.2 Effectiveness of Program 1 SCM

The combination of the strict control of cost and schedule afforded by the use of CSTAR for the Navigation Program and the technical control provided by the Program 1 SCM combined to produce a quality product within cost and schedule.

Program 1 was maintained primarily by patches to the object code rather than by updating the source code. This provided strict control, but did result in periodically having to convert a large number of patches to source code for a new baseline.

6.2.2 (continued)

This occurred approximately once a year and the new baseline was re-certified.

These methods worked satisfactorily on Program 1, partly because the program development was spread over several years.

Configuration Audits performed using the CHECKREAD capability provided clear visibility of the status of the current code.

6.3 PROGRAM 3 SOFTWARE CONFIGURATION MANAGEMENT (SCM)

6.3.1 Description of Program 3 SCM

6.3.1.1 Formal SCM

Program 3 implemented SCM as described in the representative process. The Program 3 Software Configuration Control Board (SCCB) consisted of five permanent members and other technical advisors or group representatives as needed. The permanent members were as follows:

- 1) Program 3 Project Engineer, chairman.
- 2) A member of the Systems Evaluation Department as recording secretary.
- 3) Lead programmer of Program 3 Software Development.
- 4) Lead programmer of Program 3 Software Quality.
- 5) Program 3 software coordinator.

SCCB actions required unanimous approval of all of the permanent members.

During program development Configuration Management was the responsibility of the project engineer who delegated it to the development group during code generation and to the test group during qualification testing.

6.3.1.1 (continued)

The software tool "MAPPER" was used extensively to store data, catalog, and provide status for the SCM effort on Program 3.

The ECP (Engineering Change Proposal) was used as the vehicle to initiate SCCB actions. The SCCB could approve Class II items for implementation, but all Class I items were submitted to the Program 3 prime contractor with recommendations (See section 6.1 for definitions of Class I and II ECPs).

The C&CS Configuration Management plan was furnished by the customer at the beginning of Program 3. This plan consisted of four parts:

- 1) Part I - General Configuration Management (CM) Policies and Procedures - Part I dealt primarily with the standard CM functions of identification, change control, and status accounting, as they applied to the CCS at the system level. It was oriented toward configuration management of the C&CS through control of the C&CS Specification and interface control documentation.
- 2) Part II - Interface Control Policies and Procedures - Part II addressed the policy and procedures relative to the documentation and control of C&CS subsystem to subsystem interfaces, CCS subsystem to other system interfaces, and unique CCS to LBEF interfaces.
- 3) Part III - C&CS Software CM - Part III was that portion of the plan which covered the C&CS software. It supplemented Part I by providing procedures for identification, change control, and status accounting which were specifically oriented to computer programs.
- 4) Part IV - CM at the LBEF - Part IV addressed CM activities relative to:
 - a) The shipbound equipment being tested at the facility.
 - b) The LBEF resident C&CS equipment.

6.3.1.1 (continued)

c) The test facility.

Only Part I was furnished at the beginning of Program 3. The subsystem CM plans were to be provided by the software contractor. Therefore, Sperry Univac provided the Configuration Management Plans for Programs 3 and 4. These CM Plans were required to be compatible with the system CM Plan (Part I) to ensure that subsystem changes were carefully screened for their impact at the system level, and for their proper processing when a system impact occurred.

6.3.1.2 Informal SCM

Prior to the individual placement of Program 3 configuration items under formal CM plan control, changes made in program functional and design requirements were required to adhere to the following procedure:

- 1) Pertinent data was initially documented in a meeting or phone report with copies distributed to the program manager and project engineer.
- 2) The customer representative was informed that initial investigation of other activity would take place, but that a formal written transmittal of the technical requirements and authorization to implement the changes must follow.
- 3) Change implementation or extensive technical activity preparatory to implementation was not allowed to occur until either the receipt of customer authorization or by written authorization from the project.

6.3.2 Effectiveness of Program 3 SCM

SCM on Program 3 provided the following capabilities:

- a) During software development SCM provided visibility and control through traceability of all updates and modifications of the baseline documents and program code.
- b) The Software Quality Assurance (SQA) and Software Configuration Management (SCM) groups made formal recommendations to the software development group, where feasible, to ensure the generation of quality products which met their requirements.
- c) SCM was especially valuable on Program 3 because of the large number of sites at which the multi-levels of the program were installed. The results of Program 3 were shown to be a proven acceptable software system.
- d) The SCM and SQA personnel provided an experienced pool to aid the development programmers in the areas of:
 - 1) Specification interpretation and clarification.
 - 2) Source baseline maintenance.
 - 3) Software system builds.
 - 4) Software debugging assistance.
 - 5) System integration and checkout assistance.
- e) The maintenance of a formally controlled baseline simplified speeded implementation of contractual changes of scope.
- f) Results of documentation and code audits presented in formal reports, and source baseline control by ECP provided programmer discipline and incentives to use good programming practices.

6.3.2 (continued)

- g) The strict control eased maintenance after delivery by making corrections to all copies of the system as soon as these corrections were available and tested.
- h) The SCM plans developed for Program 3 have been used as a basis for Program 4 CM and for other new projects. Each new project provided additional capabilities or refinements to the SCM Plan and procedures. This tended to shorten schedules, reduce costs, and eliminate many of the mechanical and human errors.

The main problem experienced with SCM has been the shortage of funds to adequately provide all CM capabilities. Additional help would have been advantageous for on-site CM. On-site CM personnel could have maintained the system, performed regular CHECKREADS, enforced accurate control of the program maintenance logs, enforced the use of checkout plans and procedures, and provided tighter control of the system tapes and disk packs. These activities were utilized only on a part time basis because of inadequate funding. When carried out consistently, they had a noticeable effect on the program debugging time required.

Adequate funding would also have been beneficial in controlling the source software. All updates should have been limited to and performed by a special team. This team could have been responsible for audits before and after each source update and compilation. These capabilities were only partially realized because of inadequate funding.

6.4 PROGRAM 4 SOFTWARE CONFIGURATION MANAGEMENT (SCM)

6.4.1 Description of Program 4 SCM

The SCM process implemented for Program 4 was that described as the representative SCM Plan. As an application program executing with the C&CS operating system of Program 3, Program 4

6.4.1 (continued)

shared many of the same SCM experiences as Program 3. Program 4, like Program 3 used the software tool "MAPPER" to control and locate configuration managed items in a project library. All configuration items were controlled throughout their development on Program 4 and maintained after acceptance in the project library.

Configuration control started when a programmer released a program module as debugged, tested as per his test procedure, and approved by his manager via an informal Release Notice. The magnetic tape containing the compiler directives, source code, object code, the program listing, and test procedure and the associated Release Notice were submitted to the project library for storage and control. Once the program module was in the project library and under Configuration Management Control, it was accepted by the test and evaluation group for checkout testing and integration.

Whenever it was necessary to make a change to a controlled program, the following procedure occurred:

- a) The development group documented the problem on a Computer Program Problem Report (CPPR) and prepared a Computer Program Change Order (CPCO) giving the correction and including a Program Maintenance Log showing the Object Code Changes (i.e., patch), the source library corrective code, and a listing of the affected program.
- b) The development organization manager signed the CPCO indicating approval and the change was submitted to the project library, the program listing and all documents affected by the change.

6.4.1 (continued)

Once program modules were turned over to the test and evaluation group, compilations could be performed only with the approval of the project engineer. He decided which programs required recompilation and when.

On satisfaction of checkout testing of each program module, the baseline master file was updated with the approved corrections from the CPCO and the documentation was retained in the project library. The librarian distributed copies from this baseline master for program deliveries and other activities as directed by the procurement agency.

After delivery to the procurement agency, the configuration items were still controlled and maintained in the project library, but their responsibility was transferred to the procurement agency. From that time on the procurement agency approved all changes to the baselines.

6.4.2 Effectiveness of Program 4 SCM

Program 4 was the first program to attempt to apply digital computer techniques to operations previously performed by analog computers. It was subjected to short schedules and yet it met all of its milestones and requirements and was very cost effective. This can be attributed to the top-down design concepts employed and to the excellent control provided by the configuration management procedures.

6.5 COMMON PROGRAM SOFTWARE CONFIGURATION MANAGEMENT (SCM)

6.5.1 Description of Common Program SCM

The Standard Executive Program Module MODX2 is the Navy's official baseline for the Common Program. This was furnished by the Navy to Sperry Univac to be used as the baseline for the operating system implemented in Programs 3 and 4. Configuration

6.5.1 (continued)

Control for the Common Program satisfied the principal objectives of configuration identification, configuration control, and configuration change status reporting.

The Common Program furnished to Sperry Univac early in the Program 3 cycle was modified extensively using Engineering Change Proposals (ECPs) as the method to formalize and control the necessary modifications to the code. The executive part of the Common Program was modified, recertified as the Standard Executive for the AN/UYK-7 computer, and placed under formal configuration management as a new baseline.

6.5.2 Effectiveness of Common Program SCM

The SCM procedures established for the initial ModX2 version of the Common Program have provided a documented method for incorporating modifications in the software. These procedures have been available to the Navy, other users of the Common Program, the Common Program developers, and various levels of Software Configuration Control Boards (SCCB). The Procedures required these agencies to conduct analytic reviews of each proposed software modification. The specific delineation of requirements resulting from these reviews by the concerned agencies, have clarified the overall operation and requirements of the Common Program resulting in more accurate planning. The software and documentation produced for the Common Program have also benefited from this review process. Thus, the CM procedures have made it easier to meet the program requirements within cost and schedule, and the result has been a proven acceptable software system.

The Software Trouble Report (STR) was used to document software or documentation deficiencies. The Specification Change Notice (SCN) was used to inform interested parties of changes to the

6.5.2 (continued)

approved performance and design specifications. The Engineering Change Proposal (ECP) was used to propose or record a change to the software baseline. An SCN may or may not accompany an ECP depending on whether or not specifications have changed.

STRs, SCNs, ECPs, and SCM procedures provided visibility and responsiveness for error corrections and modifications made necessary by changing customer requirements.

The review processed for SCM early in the history of the programs under study, helped expose more people in the desirability of structured programming techniques and provided an incentive to incorporate this and other good programming techniques into program development.

Use of the SCM procedures controlled the Common Program baseline and thus eased maintenance of the versions of the Common Program developed for different projects.

A problem often encountered with using SCM procedures is that costs could be markedly decreased if the procedures could allow typographical errors in documentation to be corrected without going through the full CM procedure. This does not refer to changes in content, but only obvious typographical errors and possibly minor clarifications in grammar.

SECTION 7

REAL-TIME AUTOMATIC CASUALTY RECOVERY (TASK 2.6)

7.0 INTRODUCTION

Real-time automatic casualty recovery processing was provided by software Sperry Univac developed on Programs 1, 2, and 3. The developed real-time operating system software provided this capability for third generation digital equipment, including the AN/UYK-7 computer. This software provided the capability to detect and isolate certain hardware failures, and to automatically recover real-time system operation.

The following paragraphs describe real-time automatic casualty recovery employed on the studied programs with emphasis placed on recovery from a computer module casualty.

7.1 PROGRAM 1

In developing Program 1, Sperry Univac successfully developed a software controlled real-time automatic casualty recovery capability within the constraints of the central computer complex. This capability automatically reconfigured both the digital hardware components and software components of the integrated central computer complex. The software involved in this real-time system was formed by the integration of two independent real-time applications, a navigation program and a weapons control program, under control of a shared executive operating program. The software controlling real-time automatic casualty recovery was a part of this executive operating program and affected the application programs to the extent required by the application programs to resume real-time operation.

7.1.1 Mission Requirements

The mission requirements for the central digital system included those that not only specified the normal real-time function of the computer system, but also required the system be fault tolerant, in that it:

- a) Automatically recover from all first occurrences of singular computer hardware component and peripheral failures.
- b) Maintain navigation accuracy across computer memory reloads and reconfigurations.
- c) Recovery real-time computer support in less than 4 seconds for any software reload and reconfiguration.
- d) Maintain system operation support for the mission duration, which was far in excess of the Meantime Between Failure for any single hardware component.

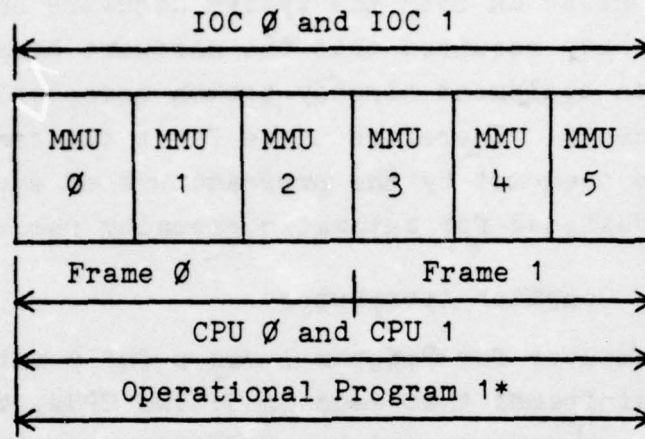
7.1.1 (continued)

These fault tolerance requirements were identified as a goal for the digital system design, production, integration and testing by Sperry Univac on both the system hardware and operational software. The Navy required that the hardware be selected from third generation equipment already proven acceptable to operational requirements. Therefore these fault tolerance requirements must have been met by the programs use of equipment not particularly designed for automatic casualty recovery.

7.1.2 AN/UYK-7 Computer Attributes

The AN/UYK-7 computer for Program 1 was a fully integrated system with two mainframes that contained; two CPUs, six 16,384 32-bit word core memories, and two IOC/IOAs (or simply IOCs) each containing 16 NTDS I/O channels. Figure 7-1 depicts this computer architecture.

Sperry Univac established that the critical functions from the central digital system could be handled by one CPU, one IOC and three MMUs. The selected configuration provided the required functional hardware components. Systems design provided for redundant or alternate peripheral hardware for each device of the central digital system. The program operating system software automatically detected singular hardware failures, isolated the failure to a hardware component, and recovered system operation by reconfiguration of both the use of hardware components and software program modules. The number of possible specific hardware configurations between the fully operational computer and minimal subset of operational functional components was very large. Only through exacting executive state software analysis, design, production, testing, integration testing, and configuration management was the real-time automatic casualty recovery of Program 1 developed and proven operational.



* - The fully operational Program 1 required as a minimum either CPU, either IOC, and any five MMUs, although normally operated with all computer resources.

The minimal operational Program 1 required either CPU, either IOC, and any three MMUs.

Figure 7-1. AN/UYK-7 Computer Architecture for Program 1.

7.1.3 Recovery Design Approach

The Program 1 real-time automatic casualty recovery was based upon the following assumptions and conditions:

- a) Hardware configuration provided a redundant or alternate component for any single component that was critical to real-time operation.
- b) Computer resident software provided a logical path to control recovery from peripheral malfunctions.

7.1.3 (continued)

- c) Computer resident software stored redundantly in main memory provided a logical path to control recovery from single occurrences of computer hardware functional component failures (CPU, MMU, or IOC).
- d) Interrupts accepted by a CPU for abnormal conditions were representative of computer hardware functional component failures, rather than software error, because the operational software had been certified as operational for each operating condition through extensive integrated system testing.
- e) Operation of the real-time program was recovered from a CPU or IOC failure by reloading software from a mass memory device and reinitializing from a copy of redundantly stored data in main memory.
- f) The C&CS Fully Operational Program required as a minimum, either CPU, either IOC and any five of the six MMUs; and the C&CS Minimal Operational Program would use, either CPU, either IOC and any three of the six MMUs.
- g) Operation of the real-time program could be recovered from a MMU failure without necessarily reloading all software.
- h) Each of the two application subsystem programs were independent such that each could operate without the other.
- i) Operation of the navigation subsystem program could be recovered in less than four seconds via reload from magnetic disk and reinitialization from parameters redundantly saved in main memory.
- j) Single occurrences of a computer hardware component failure could be detected readily by an abnormal interrupt to a CPU.

7.1.3 (continued)

These assumptions were a part of each program module responsibility for I/O with the device by providing fault detection, isolation and termination for use of a faulty peripheral device. Recovery to the redundant device, such as a second I/O channel, was also provided by each I/O handling program module. Whereas, selection of an alternate device was the responsibility of the program module requesting input or output of information via the I/O handling program modules.

Real-time automatic casualty recovery of a computer component failure was the responsibility of special purpose executive software and processor firmware (programs in NDRO memory).

This software and firmware included:

- a) On-line, Confidence Tests - tests detecting malfunctions in CPUs, MMUs or IOCs. (Refer to the section titled "Real-time On-line Maintenance.")
- b) Isolation Tests - tests run after detection of a suspect failure to identify the failure within a CPU, MMU or IOC.
- c) Error Processing - routines which control fault isolation and operational program recovery for abnormal CPU interrupts.
- d) Memory Resume Processing - redundantly stored recovery routines to reconfigure/reinitiate software such that a MMU generating the resume was suspended from use by the operational program,
- e) Recovery Routines - routines to provide for reloading of software as necessary, reinitialization of suspended program modules and recording of the assumed fault condition for later manual maintenance testing and repair.

7.1.3 (continued)

As part of Program 1, on-line CPU confidence tests were run periodically in the background mode during dual CPU operation. This program attempted to "halt" any malfunctioning CPU executing the test. This "halt" left the remaining CPU to detect, via software, that it was the only one operating and would then call upon the CPU recovery routine. Other confidence tests in Program 1 which would test memory addressing, IOC control memory and CPU arithmetic processing were directly executable as subroutines by any program module. For example, either resident control program module of the two application subsystem programs would run these tests before initiating operational use of the tested computer resource. If any test detected a malfunction, it identified the faulty hardware component and called for the appropriate recovery routine.

Isolation tests were called by Error Processing to logically select the probable fault hardware component. Isolation tests included:

- a) The directly executable on-line confidence tests.
- b) The hardware component interface test provided by NDRO (CPU to IOC, CPU to MMU and IOC to MMU).
- c) An error proximity memory data validity test.

The Error Processing software routines used these isolation tests to locate a failure within a hardware component. If these tests were unable to detect a failure, Error Processing assumed that the interrupt was either a non-test detectable hardware component failure, CPU failure for Class 2, program faults or IOC failure for Class 3, I/O faults) or a degeneration of program stored in main memory.

7.1.3 (continued)

Therefore, when a redundant hardware component was available, the suspect component was reconfigured logically from operational use and the operational software reloaded, and program operation was resumed via the recovery routines. The Error Processing firmware routine (Hardware fault Analysis Routine in NDRO) processed all Class 1 hardware faults, except for a Power Tolerance interrupt. This routine sorted memory resumes for processing by software in a responsive MMU other than the resuming MMU.

See figure 7-2 for a description of the recovery routine's processing provided on Program 1 for each class of interrupts.

Program 1's automatic casualty recovery design approach was primarily to maintain computer support of the navigation equipment even though casualty processing sometimes required reloading the operational software. Reloading was possible because the allowable suspension of supporting the navigation equipment was between 3.5 and 4.0 seconds. The program stored key navigation data in computer memory in real-time and periodically updated other data on disk memory. This provided a full data set after the software reload and allowed the navigation program to continue execution from established check points. This approach secondarily provided for resuming the less time-critical application subsystem program.

<u>TYPE OF INTERRUPT</u>		
<u>CLASS</u>	<u>SUBCLASS</u>	<u>NAME</u>
I:	1.	CP-OPERAND MEMORY RESUME
	2.	CP-IOC COMMAND RESUME
	3.	CP-INSTRUCTION MEMORY RESUME
	4.	CP-IOC INTERRUPT CODE RESUME
	5.	IOC-MEMORY RESUME
	6.	INTERCOMPUTER TIMEOUT (QUEUED)
	7.	POWER TOLERANCE (NEVER DISABLED)
II:	1.	INTERPROCESSOR INTERRUPT (QUEUED)
	2.	FLOATING POINT ERROR
	3.	ILLEGAL INSTRUCTION
	4.	PRIVILEGED INSTRUCTION ERROR
	5.	OPERAND BREAKPOINT MATCH
	6.	OPERAND READ OR INDIRECT ADDRESS
	7.	OPERAND WRITE
	8.	OPERAND LIMIT
	9.	INSTRUCTION BREAKPOINT MATCH
	10.	INSTRUCTION READ
	11.	INSTRUCTION LIMIT
	12.	MONITOR CLOCK (QUEUED)
III:	1.	IOC ILLEGAL INSTRUCTION (QUEUED)
	2.	IOC ILLEGAL CHAIN INSTRUCTION (QUEUED)
	3.	IOC-MONITOR CLOCK (QUEUED)
	4.	IOC-PROCESSOR INTERRUPT (QUEUED)
	5.	EXTERNAL INTERRUPT MON. (QUEUED)
	6.	EXTERNAL FUNCTION MON. (QUEUED)
	7.	OUTPUT DATA MON. (QUEUED)
	8.	INPUT DATA MON. (QUEUED)
CLASS IV	1.	

Figure 7-2. Software Interpretation of AN/UYK-7
CPU Interrupts (Sheet 1 of 7)

PROGRAM 1

Programmed Automatic Isolation.

Interrupt representative of:

- a) MMU Malfunction:
 - . Class I, Subclasses 1, 3, 5.
- b) IOC Malfunction:
 - . Class I, Subclasses 2, 4, 6.
 - . Class III, Subclasses 1, 2.
- c) Power Failure:
 - . Class I, Subclass 7,
i.e. Power Tolerance when followed by a Master Clear
Auto Start.
- d) CPU Malfunction:
 - . Class II, Subclass 2, 3, 4, 6, 7, 8, 10, 11

Interrupt conditions not shown above were processed as a normal event and not included in casualty considerations with the exception when one was not expected (e.g., a monitor interrupt on a IOC channel not in use). For these exceptions confidence tests were performed to detect any hardware component failures and a reload/reinitialization was provided to the real-time program to restore software integrity.

Figure 7-2. Software Interpretation of AN/UYK-7

CPU Interrupts (Sheet 2 of 7)

PROGRAM 1

Programmed Automatic Recovery:

a) MMU Malfunction:

If the MMU indicated in the interrupt code contained the operating systems software, the realtime program was reloaded and configured around the MMU flagged as "unusable", and reinitialized. If the MMU indicated contains only an application program, the affected program was suspended, the MMU was flagged "unusable", the spare memory area was assigned to the affected application program, and a warm start was signed to the application program's resident control module.

b) IOC Malfunction:

The IOC indicated in the interrupt code is flagged as "unusable" and a message was transmitted to each application program's resident control module.

c) Power Failure:

The auto start software in each CPU checked for interfaces to formerly usable IOCs and MMUs, flagging any inaccessible components as "unusable" reconfigured the memory allocation, reloaded and reinitialized the real-time program.

d) CPU Malfunction:

Error processing attempted to stop the CPU to prevent its continued use. The remaining CPU detected the absence of the stopped CPU, flagged the stopped CPU as unusable, freed all possible dedicated resources and continued real-time processing.

In all above cases, the operator's console was sent descriptive messages of the change in component status. For single component isolations, the operating system software provided a means to run the AN/UYK-7 diagnostic program and to manually repair a failed component.

Figure 7-2. Software Interpretation of AN/UYK-7 CPU Interrupts (Sheet 3 of 7)

PROGRAM 2

Interrupts suspected to detect hardware malfunctions were:

- . Class I Subclasses 1, 2, 3, 4, 5, 6 when illogical and 7 when followed by a master clear auto start.
- . Class II Subclasses 3, 4, 6, 7, 8, 10, 11, and 5, 9 and 12 when illogical.
- . Class III Subclasses 1, 2, and 3, 4, 5, 6, 7, and 8 when illogical.
- . Class IV When the executives software appeared to have failed.
 1. Terminated the real-time operational program.
 2. Loaded and executed the System Loader to:
 - a. Execute comprehensive confidence tests on:

IOC - command address registers non-buffered timing, and interfaces to CPU and MMUs. (Any error implies an IOC malfunction).

MMU - addressing logic, and data validity, and interfaces to CPU.

CPU - each of the two C&CS CPUs operate the IOC and MMU tests one after another, if the first CPU does not complete in the allowable time, it is assumed as failed. Any unexpected condition occurring during the further process of reconfiguration, reloading and reinitialization resulted in a halt of the operating CPU.
 - b. Reconfigured the memory map and allowable real-time functions, reloaded the resident portion of the C&CS Base Program and provided the real-time initialization for a reload.

Interrupt conditions not listed above were processes as normal events.

Figure 7-2. Software Interpretation of AN/UYK-7
CPU Interrupts (Sheet 4 of 7)

PROGRAM 2

For component malfunction isolated by the System Loader tests, the System Loader provided for:

a) IOC malfunction:

System Loader flagged the IOC as unusable for the operational program. (All CP/IOC, MMU/IOC, and I/O channel status flags were set non-operational for the IOC.)

b) MMU malfunction:

System Loader flagged the MMU as unusable for the operational program.

c) CPU malfunction.

System Loader attempted to Halt the CPU via response to an interprocessor interrupt and flagged the CPU as unuseable for the operational program.

If available, the second CPU attempted to process reconfiguration, reloading and reinitialization of the Base Program.

Automatic recovery was complete when the C&CS Base Program was again operating.

Figure 7-2. Software Interpretation of AN/UYK-7

CPU Interrupts (Sheet 5 of 7)

PROGRAM 3*

Programmed Automatic Isolation:

Interrupt representative of:

a) MMU Malfunction:

- . Class I, Subclasses 1, 3, 5

b) IOC Malfunction:

- . Class I, Subclasses 2, 4

c) Power Failure

- . Class I, Subclass 7

i.e. Power Tolerance when followed by a Master Clear
Auto Start.

d) Intercomputer I/O Malfunction

- . Class I, Subclass 6

e) Software Malfunction

- . Class II, Subclasses 3, 4, 5,**6, 7, 8, 9,**10, 11, 12**

- . Class II, Subclasses 1, 2, 3,**4,**5,**6,**7,**8,**

* - Program 3 provided the operating system for Program 4.

** - Only a malfunction when no program module is registered to
normally process the interrupt.

Figure 7-2. Software Interpretation of AN/UYK-7

CPU Interrupts (Sheet 6 of 7)

PROGRAM 3

Recovery Resulting By:

a) MMU Malfunction:

The identified MMU was flagged as failed and unusable for the operational program. Normal operation of the local computer was suspended, the remote computer informed of the failure and transition to a degraded configuration was made by loading and executing the Software Configuration Module which reloads, initializes and starts the degraded configuration. A partial load was simultaneously performed in the remote computer.

b) IOC Malfunction:

The identified IOC was flagged as unusable for the operational program, all channels affected were flagged unuseable and all registered channel using regional programs were sent a notification message.

c) Power Failure:

Response of IOCs and MMUs to each on-line CPU was tested. If the online components all respond correctly, than a "warm" restart was given to all former on-line program modules. If a component does not respond, it was flagged as failed and proper component malfunction recovery taken.

d) Intercomputer I/O Malfunction.

Communication was reestablished on the redundant I/O channel. If communication was impossible to establish, the software was reconfigured to assume total failure of the remote computer.

e) Software Malfunction

If in the Operating System program, the software was fully reloading. If in a regional programs control module, the regional program was reloaded. If in a non-control module if a regional program, then the region's control module was notified of the faulty software module.

Figure 7-2. Software Interpretation of AN/UYK-7
CPU Interrupts (Sheet 7 of 7)

7.1.3 (cont.)

The key to fault detection was that any "abnormal" interrupt or test detected malfunction was assumed to indicate a hardware failure. Through testing, the failure was isolated to a hardware component. By recovery, the failed component was deleted from operational use, software was reconfigured and operation restarted. Finally, a notification was provided to be descriptive of the failure and new operating condition for later manual analysis and repair.

7.1.4 How Success Was Achieved

Univac established the basic concept and presented its feasibility to the Navy. There was a continuously working relationship between Sperry Univac's project management and the Navy agency responsible for the C&CS program. Project management understood and informed technical personnel just what the Navy requirements were and informed the Navy of development status through meetings, reports and phone calls. Under Navy direction, system design reviews were frequently held during analysis, design, production, testing and integration of the C&CS program.

The analyst began with the known system requirements, and with support from knowledgeable hardware engineers, reviewed the system requirements and established a hardware design to meet these requirements. From this, the requirements on software were established. As a part of this analysis the sequence and timing of system functions was predicted and interpreted as software functions, sequence, priority, timing and usage of CPU time, I/O channel usage and main memory allocation. Recently the accuracy of this analysis was proven by a collection and analysis of empirical data taken on the final operational program. This data closely matched these earlier reported predictions for CPU loading, I/O loading and memory utilization.

7.1.4 (cont.)

For the most part, each programmer worked individually on a program module under supervision of a lead programmer or section head and in accordance to the written Common Program Guidelines. Therefore, each programmer was responsible for specification design, coding, debugging and testing support of one or more modules.

The key to success on this project was its personnel manager's analysts, and programmers worked as teams throughout the effort. The engineers each worked for a long period (six months and longer) of the schedule. These engineers offered knowledge in use of the hardware and experience in providing the software required to complete this program.

The software techniques shown in the final shipboard program model did not come about quickly or easily. These techniques were established by many hours of actual use on the computer test system by many engineers. Some modifications were required in hardware and in firmware to succeed in producing this operational real-time casualty recovery. The successes shown during the seven years spent finalizing this development affected the design of simultaneous and future developments. Affected system developments included Programs 2 and 3, as well as, the Navy configuration controlled baseline Common Program.

7.2 PROGRAM 2

7.2.1 Mission Requirements

Program 2 development was part of the large integrated ship-board Command and Control System (C&CS) operational program. This C&CS program provided ship command personnel with the capability to continuously monitor the ships tactical mission environment and to control its weapon systems. C&CS generalized mission requirements included the following:

- 1) The C&CS operational program must continuously support a wide range of activity levels and must be ready to maintain real-time operation for an extended period.
- 2) The C&CS program structure and operating system software must allow operator entries to efficiently reconfigure the on-line software for varying conditions of operational readiness within the available operable equipment.
- 3) The C&CS operational program must provide a tolerance of fault conditions by malfunction or failure of any computer complex hardware components which include the functional modules of the AN/UYK-7 computer (CPUs, IOC/IOAs, and MMUs) and interfaced devices.
- 4) The software providing real-time automatic casualty recovery must minimize the use of computer main memory.

These requirements were met by the computer complex hardware design and software capabilities.

The following paragraphs concentrate on the real-time automatic casualty recovery of malfunction or failure of a functional component of the C&CS AN/UYK-7 computer.

7.2.2 AN/UYK-7 Computer Attributes

Program 2 operated within a four mainframed AN/UYK-7 computer containing three CPUs, thirteen MMUs each containing 16,384 32-bit words of core memory, and four IOC/IOAs each containing 16 NTDS I/O channels. Figure 7-3 depicts the computer architecture showing the inter-connections that were available to the two operating system programs using this computer, i.e. C&CS operational program and Sensor System Program. Since the C&CS operational program was critical to the ship's mission, the allocation of computer components to the C&CS program dominated the allocation of components to the Sensor System Program. The C&CS program normally utilized CPU 0 and 1, MMU 0 through 8, and IOC/IOA 0 and 1. The remaining components were normally utilized by the Sensor System Program. Both operational programs shared specified areas of MMU 8 and 9, and specified channels on IOC/IOA 2 and 3. The operating system software of both programs provided the necessary control and communication for use of these shared components.

Although the operating system of the Sensor Program did provide tolerance of malfunction and failure of device external to the computer, and did operate without use of MMU 8, any failure of CPU 2, or MMUs, 9, 10, 11 or 12 resulted in termination of operation of the Sensor System Program. After termination of the Sensor System, the C&CS controlling system operator may have reconfigured the C&CS software and operated either of the two major Sensor functions as part of the C&CS operating configuration.

This computer architecture supported continuous operation of the C&CS operational program for many varied configurations upon occurrence of a malfunction or failure of one or more functional components. Since the operable computer components

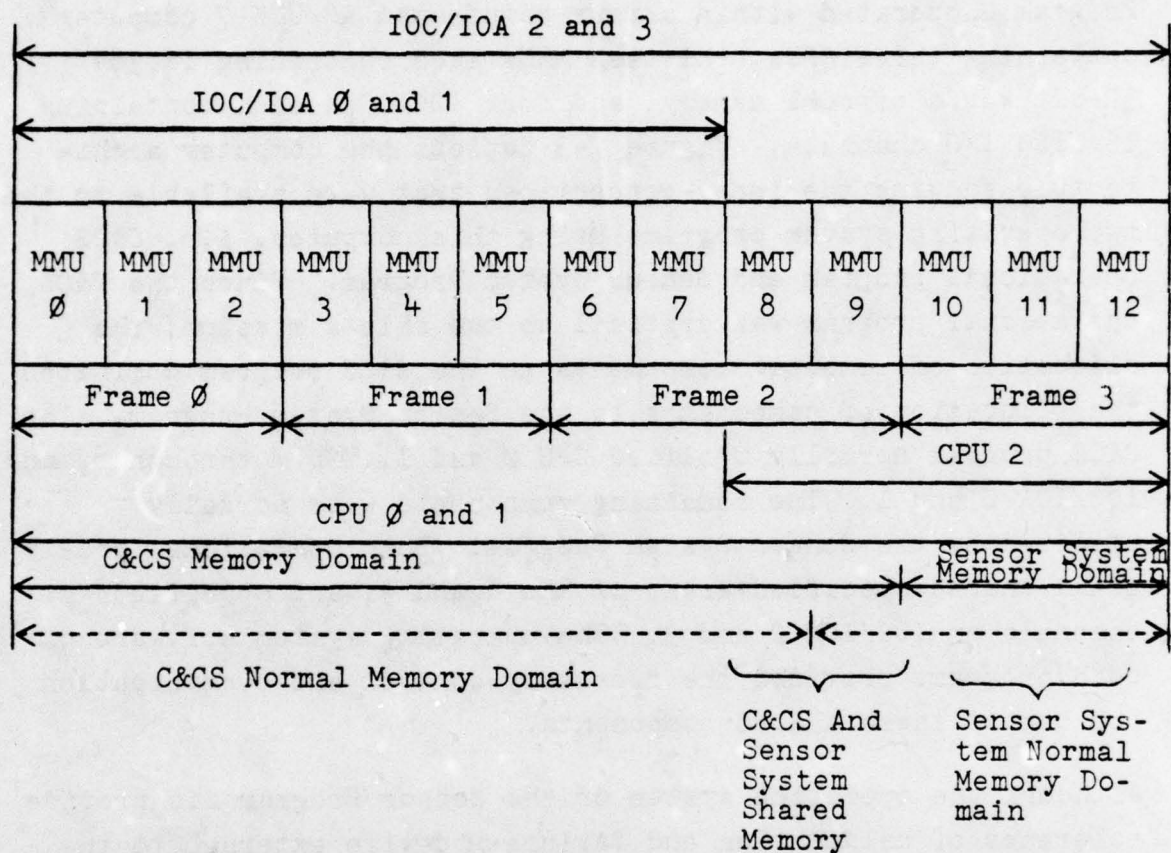


Figure 7-3. AN/UYK-7 Computer Architecture For Program 2

7.2.2 (cont.)

required for minimal C&CS support (i.e., operation of the C&CS base program) was one CPU, one IOC/IOA and six MMUs, the number of possible configurations selected from two CPUs, four IOC/IOAs, and thirteen MMUs was large.

7.2.3 Recovery Design Approach

The software that controlled real-time automatic casualty recovery after failures occurred within the C&CS central computer complex was provided by three subprograms of the real-time operating system called System Leader, Dynamic Reconfiguration, and Common System Error Processing. Together these subprograms provided the capability to load the operational program into the C&CS computer, to reconfigure the C&CS Operational Program in real-time, and to identify error conditions representative of a computer hardware failure. These subprograms also provided automatic isolation of a computer functional component containing a detected failure from further operational program use and provided reconfiguration of the operational program.

The System Loader subprogram was the first program to be loaded into the C&CS computer. It was the only part of the C&CS Operational Program to be loaded via the AN/UYK-7 NDRO bootstrap program. When loaded, the System Loader performed those tasks required to load and initialize the C&CS resident base program (a minimal subset of Program 2). These tasks consisted of:

- a) Establishing an interface with the Sensor System programs.
- b) Performing comprehensive diagnostic tests to determine the operability of the C&CS computer functional components and mass storage devices (magnetic tape and management disk).

7.2.3 (cont.)

- c) Allocating main memory for use by the resident base program.
- d) Obtaining control of the mass storage device.
- e) Initializing the system controlling tables (segment directory, function directory, module directory, and segment base address tables).
- f) Initializing and activating the base program.

The System Loader provided the system controlling tables to the Dynamic Reconfiguration subprogram for use during the real-time operation.

Four memory areas contained a routine from the System Loader to automatically reload the System Loader via the NDRO bootstrap program whenever a MMU failure was suspected. Also, whenever the Common System Error Processor identified an error condition representative of a hardware failure, the Error Processor suspended real-time program operation and initiated software reloading by calling one of these routines to bootstrap the System Loader. This automatic casualty recovery provided minimal use of main memory because the System Loader interface data and reloading control routines were not computer resident during real-time operations.

The Dynamic Reconfiguration subprogram was responsible for the efficient management of the available memory resources (main memory, and mass storage) with respect to the needs of the operational program object elements. Dynamic Reconfiguration interfaces the controlling system operator and real-time subprograms performed the functions required for task program loading, reconfiguration and modification in real-time. Dynamic Reconfiguration was contained within the C&CS base program and therefore was active after being loaded and initialized by the

7.2.3 (cont.)

System Loader. Dynamic Reconfiguration used the system controlling tables to perform the following functions in real-time:

- a) Provide program load control.
- b) Control program module activation/deactivation.
- c) Allocate/deallocate memory,
- d) Maintain computer memory mapping.

The C&CS Operational Program used these functions to provide the general capabilities to:

- a) Load the C&CS Operational Program transient task modules in preselected configurations.
- b) Change the existent software configuration automatically in response to other software requests or in response to operator action.
- c) Load programs which were not normally part of the C&CS operational configuration such as hardware test or diagnostic programs (see section 8.1).

The range of program elements which could be loaded or deleted were:

- a) Program segment - basic element of a program module, such as an instruction or a data segment which could be a transient segment of an otherwise resident program module.
- b) Program Module - included all segments required to accomplish processing of a function.
- c) Program Configuration - all modules related to a tactical situation or state of operational readiness.

7.2.3 (continued)

The range of program elements which could be loaded or deleted were:

- a) Program segment - basic element of a program module, such as an instruction or a data segment which could be a transient segment of an otherwise resident program module.
- b) Program Module - included all segments required to accomplish processing of a function.
- c) Program Configuration - all modules related to a tactical situation or state of operational readiness.

Dynamic reconfiguration was necessary to real-time automatic casualty recovery because only the C&CS base program was loaded and activated by the System Loader, and the capability for program reconfiguration was provided by Dynamic Reconfiguration. Dynamic Reconfiguration provided a function to predetermine the possible configurations allowable from the available computer resources. The operator at the System Control display console was provided selection of the correct possible basic configurations for each condition of readiness and the allowable additions or deletions of operable functions within each configuration.

The Common System Error Processing subprogram provided the real-time control of the C&CS computer resources in the event of an AN/UYK-7 CPU interrupt identified as abnormal. Error Processing sorted these events as recoverable and provided the software to control recovery by calling for the reload of the System Loader which then loaded and initialized the C&CS base program. Refer to Figure 7-2 for a description of the Program 2, Error Processing of AN/UYK-7 CPU Interrupts.

AD-A040 049

SPERRY UNIVAC ST PAUL MINN DEFENSE SYSTEMS DIV

F/G 9/2

MODERN PROGRAMMING PRACTICES STUDY REPORT.(U)

APR 77 W E BRANNING, D M WILLSON

F30602-76-C-0136

UNCLASSIFIED

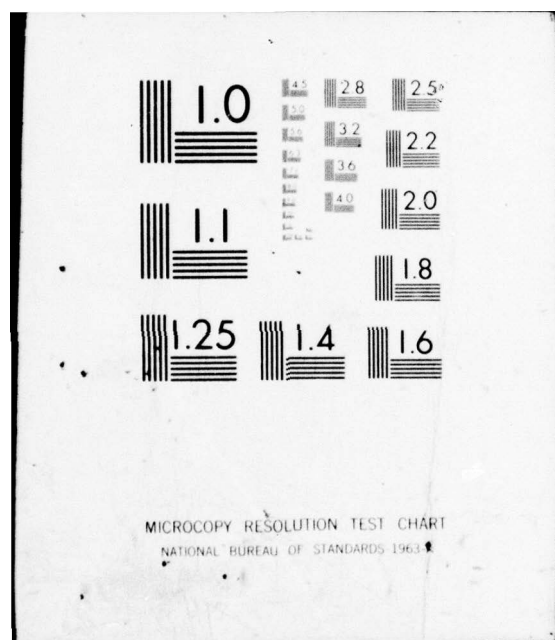
PX-11932-F

RADC-TR-77-106

NL

4 OF 5
AD
A040049





7.2.4 How Success Was Achieved

As already described in the sections on Program Environments and Development Standards, the Program 2 real-time operating system began development from a certified baseline Common Program. Real-time automatic casualty recovery was an operating system capability added to this baseline software. Success was achieved by optimizing the operating system software recovery functions to:

- a) Those of real-time error processing.
- b) Real-time dynamic operational readiness control.
- c) Hardware fault detection testing, fault isolation and recovery.

These functions fulfilled the mission requirements for real-time automatic casualty recovery suitable to this C&CS.

The program was modular and the ground rules for establishing each program module were defined in a study report (see appendix C) before software specification and design began. These ground rules were strictly enforced by using the Common Program as the starting point to the operating system and by using SYSMAKER for the system tape generator.

The detailed analysis of intermodule dependencies was interpreted by tracing all system functions and listing interdependent sets of program modules. These sets of program modules were assigned by the System Loader and Dynamic Reconfiguration programs as program modules required to be computer co-resident for each possible condition of operational readiness. The System Loader isolated the failed component, reconfigured the use of available

7.2.4 (continued)

resources and reloaded and restarted operation for the C&CS base program. When operation of the base program was resumed, the system control operator configured to the appropriate condition of operational readiness and selected operation of the desired addition functions which could be supported by the available system resources.

7.3 PROGRAM 3

Real-time automatic casualty recovery was a major part of the software development effort on Program 3. Three elements of the C&CS operating system software (the Software Configuration, the Reconfiguration Routines and Common Error Processing) provided the software for automatic casualty recovery. Additionally, these elements provided a versatility to use the hardware resources in a planned manner for numerous operational configurations. Operational configurations were selected automatically in response to conditions of hardware malfunctions and were selectable by the system operator for manual software reconfiguration. Similar to Programs 1 and 2, these configurations assured support of minimal requirements critical to the ship's mission. The fundamental differences between the Program 3 software configurations Programs 1 and 2 was that:

- a) Program 3 supported a greater number of application subsystem programs, five in all.
- b) The central computer was two multiple processor computers under control of one operating system software package.

7.3.1 Mission Requirements

Program 3, when integrated with the Navy-furnished Common Program, provided the real-time operating system software for a large central digital C&CS operational program. The C&CS operational program formed the digital control for the ship's sensors, weapons, and control systems. All of these functions were critical to the ship's mission.

The mission requirements of the ship were interpreted as required capabilities and functions of the C&CS hardware design and operational program. The analysis of these requirements and system design resulted in forming the required capabilities and

7.3.1 (continued)

functions to be available in several modes of the operating configuration (fully operational, by pass, degraded, failed and recovery). The Program 3 contracting agency conducted this system analysis over a period of two years and produced seven revisions of a resulting study report before finalizing the C&CS system operational system configurations in the Degraded Mode Report. This final report presented overall design and with Navy approval was made a part of the C&CS design requirements. Thereby, the shipboard requirements were provided as direction to Sperry Univac for the operational system configurations and imposed operating system software requirements supporting these configurations.

This system multiple configuration design required that a single operating system program provide the following:

- a) One of two distinct subsystem application programs be selected for three of the five C&CS application programs to provide full capability or degraded capability.
- b) The two remaining C&CS application programs be operational in one or the other AN/UYK-7 computer, (Program 4 was one of these application programs), and
- c) That the operating system program provide system control logic, system interfaces, and operator interfaces to establish and transition between twelve defined operational configurations.

While accommodating C&CS application subsystem program configurations, the C&CS operating system was required to accommodate hardware component failures affecting the central digital system operation. Hardware components included:

7.3.1 (cont.)

AN/UYK-7 components, peripheral equipments and ship's services such as electrical power. The system configurations were designed for recovery from each of these failures for singular failures and for specific combinations. The configurations not only provided for recovery from anticipated component failures, but also provided for the maintenance necessary to restore the system to the fully operational configuration.

System analysis on Program 3 was supported technically by Sperry Univac engineering and established that the mission critical requirements could be sustained for the mission duration by a properly integrated hardware/software design.

7.3.2 AN/UYK-7 Computer Attributes

Program 3 operated within the central computer complex consisting of two AN/UYK-7 computers called Computer A and Computer B, and peripheral devices including:

- a) Magnetic Disk Memory - two units, each duplexed to each computer.
- b) Magnetic Tape Unit - a two drive unit, duplexed to each computer.
- c) Signal Data Converter - two units, each duplexed to each computer.

Computer A contained the following functional hardware components in three integrated mainframes:

- a) Two CPUs - each totally integrated with the MMUs and IOCs.
- b) Eleven MMUs - five pairs of interleaved addressable memory units of 32,769 32-bit words and one non-interleaved memory of 16,384 words.

7.3.2 (continued)

- c) Two IOC/IOAs - or simply IOCs, each IOC was totally integrated with the CPUs and MMUs, and provided 16 NTDS I/O channels each.

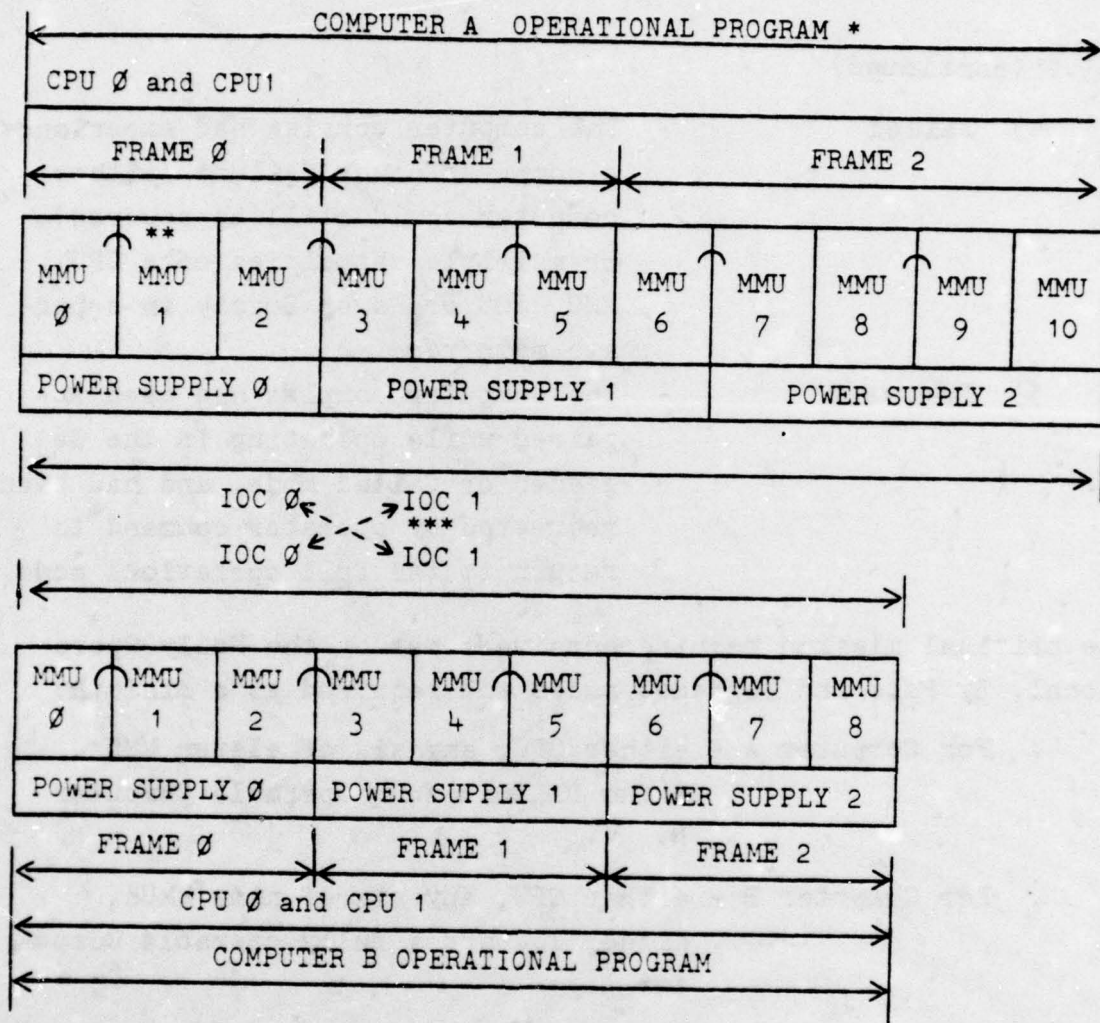
Computer B contained the following functional hardware components in three integrated mainframes:

- a) Two CPUS - each totally integrated with the MMUs and IOCs.
- b) Nine MMUs - four pairs of interleaved addressable memory units and one non-interleaved unit.
- c) Two IOC/IOAs - or simply IOCs, each IOC was totally integrated with the CPUs and MMUs, and provided 16 NTDS I/O channels each.

Computer A and Computer B were interconnected by two intercomputer I/O channels. Figure 7-4 depicts this computer system architecture. All channel cabling was assigned such that redundant or alternate devices were connected to separate IOCs.

The degraded mode analysis on this C&CS presented five basic operational modes as follows:

- 1) Fully Operational- all computer complex hardware was operational and all subsystem programs were providing full capability support.
- 2) By Pass - The computer complex had experienced a peripheral, I/O channel or an IOC casualty.
- 3) Degraded - The computer complex had experienced a CPU, MMU, or Power Supply casualty.



* - The operational program for computers A and B operated within twelve distinct modes.

** - \cap shows interleaved MMU pairs.

*** - \longleftrightarrow shows intercomputer I/O channels.

Figure 7-4. AN/UYK-7 Computer Architecture For Program 3 (and 4)

7.3.2 (continued)

- 4) Failed
 - The computer complex had experienced a computer power failure (either computer could still be powered), or multiple casualties of a CPU, MMU, IOC or Power Supply in separate mainframe.
- 5) Recovery
 - The computer complex had been repaired while operating in the degraded or failed mode, and had been requested by operator command to return to the full operational mode.

The critical mission requirements were met by the Fully Operational, By Pass and Degraded modes and required as a minimum:

- . For Computer A - either CPU, any six of eleven MMUs, either IOC and a fully operable Computer B.
- . For Computer B - either CPU, any six of nine MMUs, either IOC and a fully operable Computer A.

The C&CS was required to continue real-time operation even after multiple failures to a minimum level of support by one computer with components operable in two of its three mainframes. Therefore, the number of different possible operable hardware configurations was greater than either those experienced by Program 1 or 2.

7.3.3 Recovery Design Approach

The overall analysis to provide the C&CS operational configurations that would result from automatic real-time casualty recovery was presented to the Program 3 development group by the customer. This analysis specified twelve specific software

7.3.3 (continued)

configurations to be automatically selected by Program 3 as diagrammed in figure 7-5. The Program 3 development group created three major software elements to automatically assume or transition to the specified program configuration as a result of a computer complex component casualty or by command from the system operator.

The three software elements were as follows:

- 1) Software Configuration (SC) - SC was a loader program that upon initial software load, or recovery reload, established the computer operational configuration, selected the software configuration, and loaded and initialized the C&CS resident operational program modules defined to implement the selected configuration.
- 2) Reconfiguration Routines (RR) - RR was a computer resident element of the C&CS operating system software which monitored the system status of both computers, controlled changes in system status, and controlled real-time transition from one operational configuration to another.
- 3) Common Error Processing (CEP) was a computer resident element of the C&CS operating system software which directed fault isolation and recovery upon the occurrence of any abnormal CPU interrupt.

Software Configuration (SC) was computer resident only during initial loading and automatic reloading of the C&CS operational software. SC was designed to contain and execute a sequence of tests upon the computer complex components including:

BEST AVAILABLE COPY

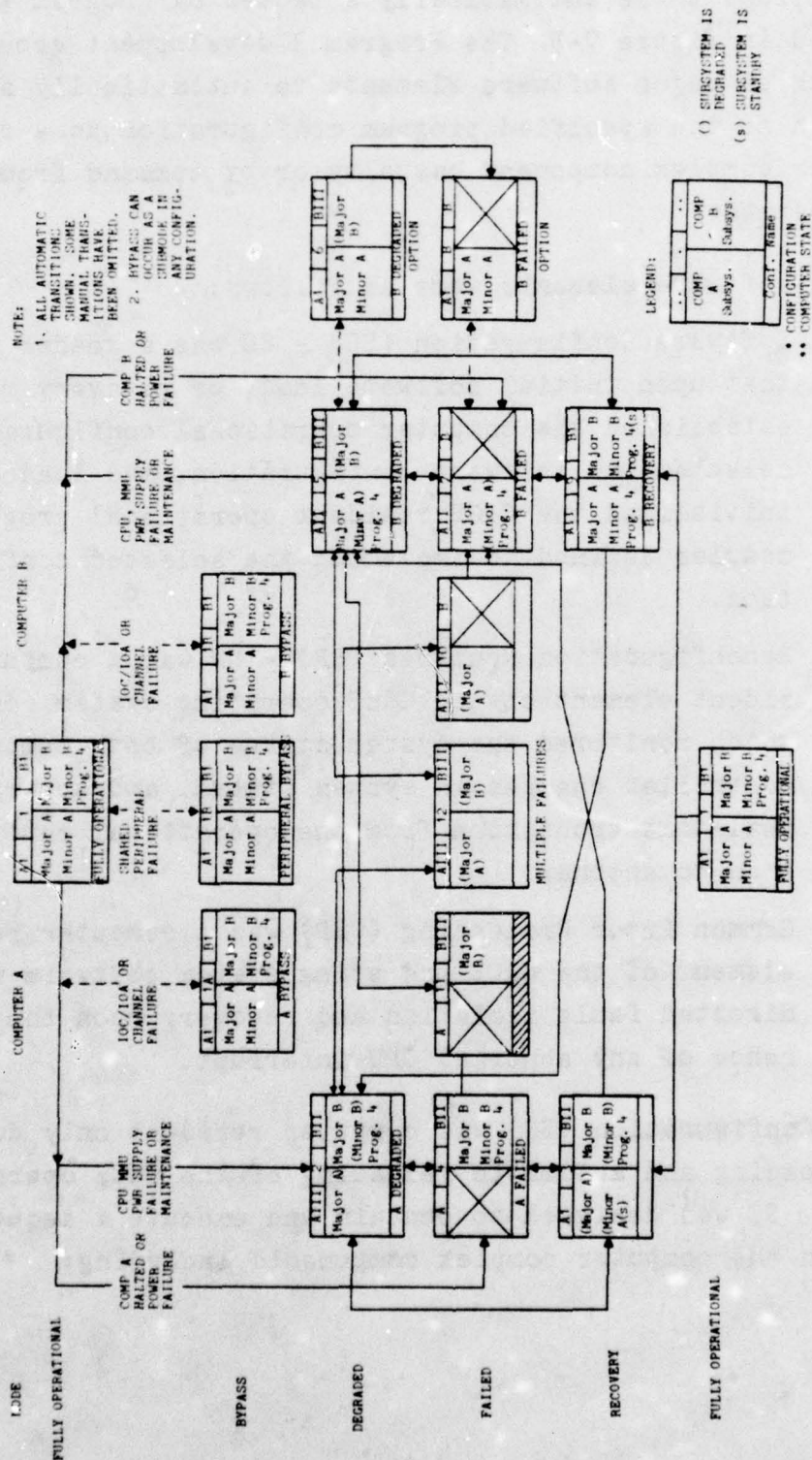


Figure 7-5. Central Computer Complex Operational Configurations and Transitions Diagram

7.3.3 (continued)

- a) Component Interface Tests - Tested interfaces; CPU to IOCs, IOC to MMUs, CPU to MMUs for each CPU, each IOC and in both computers. Tested I/O interfaces to the magnetic disk or tape for operational program loading.
- b) Component Confidence Tests - Tested operation of each CPU, IOC and MMU sufficiently to select a set of units operable for loading a configuration of the C&CS operational software.
- c) Monitor tests - Each CPU consecutively performed tests a) and b) while the other CPU waited and timed the CPU running the tests. If the CPU running the tests exceeded a timeout value, the CPU was flagged as unusable and the monitoring CPU began as the only available CPU. Also, SC timed all input actions to detect any failure while loading the operational software.

As a result of the former system status of both computers and the result of these tests, SC established proper operational program, loaded the resident portion of this program, initialized the loaded program, and transferred CPU control to initiate real-time operation.

Upon initialization, the Reconfiguration Routines reaffirmed the operation configuration by exchanging system status information with the other computer. If RR determined that the initializing programs configuration was not compatible, RR selected the proper configuration and called for SC to reload the operational software. When RR determined that the software configuration was correct, RR allowed real-time initialization to be completed.

7.3.3 (continued)

This initialization included RR establishing a monitor-other-computer function to detect any future change in system status in either computer.

RR also provided internal software interfaces for the resident control module of each subsystem application program, to the Common Error Processor and to the system operator. Upon demand, these interfaces returned definition of the system status (critical data that was saved in main memory during a full software reload), and provided for changes to system status. Upon any change to system status, RR directed the transition indicated by the new operational configuration. These transitions consisted of:

- a) Exchanging fully capable subsystem program with their degraded capability program and visa versa.
- b) Removing one subsystem program from one computer while loading and initializing that subsystem program in the other computer.
- c) Calling for an automatic full software reloading of either computer.

Common Error Processing was a collection of computer resident routines that directed isolation of computer complex components suspected of failure (indicated by an abnormal CPU interrupt), and directed automatic recovery of the operational program by a coordinated operation with RR and each subsystem programs' resident control module. See Figure 7-2 for a description of the processing provided for an abnormal CPU interrupt.

7.3.3. (continued)

Program 3 provided for automatic casualty recovery for a computer or peripheral functional component malfunction by terminating use of the suspect component by the operational program until the system operator indicated the component was operational. According to status of all functional components for both computers and the status of the operating software, a change in any item of status resulted in Program 3 action to establish the proper software point for the Navy-controlled baseline Common Program. Since this Common Program was the starting point for this operating system, much design was provided for automatic casualty recovery, especially for the Common Error Processing element. The Program 3 design for automatic casualty recovery was established in the preliminary performance and design specifications. The produced software was revised each time the Degraded Mode Report was reissued. This development process resulted in the final development model of Program 3. After considerable review, the contracting agency determined that RR would be redeveloped for the sea test model of Program 3.

This sea test model of RR was designed to be logically driven by tabled data, rather than by in-line program logic and, therefore, would be more easily modified during integration with the final computer complex hardware and final subsystem application programs. This design allowed for automatic operation recovery from singular and multiple failures, on-line failure diagnosis and repair, and returning of components to operational status.

7.3.4 How Success Was Achieved

The contracting agency for Program 3 was also the C&CS integration agency. Therefore, the contracting agency was able to establish the requirements for the operating system software

7.3.4 (cont.)

and application subsystem software such that the system functions critical to the mission could be maintained for any single component failures, and optimum support of mission critical functions could be provided for many multiple component failures. Sperry Univac engineering had already proven that real-time automatic casualty recovery of computer functional components was attainable on Program 1.

All contracting agencies worked with the integrating agency and underoverall Navy management to establish automatic casualty recovery. As a result, Sperry Univac developed the sea test model of Program 3, qualified by test that it did automatically establish each specified system configuration, and did function as the operating system for the two multi-processor computer complex.

7.4 CONCLUSIONS

Real-time automatic casualty recovery of operational programs by software control was attained as part of developing Programs 1, software control was attained as part of developing Programs 1, 2, and 3. Each development required a proper hardware design that allowed in reserve, or through reconfiguration, resources that were reassigned when other resources were not available. These replacement resources were provided by redundant and alternate functional hardware components which were electronically isolable one from another and accessible for reconfiguration by software.

In each program, specifically designed elements of the operating system provided interpretation of mission requirements for the optimum use of the usable hardware components. The software structure was modular, and modules combined to form subsystem programs, such that the operating system could establish any software configuration by specifying a list of program modules. Additionally, the operating system software centralized the detection of faults, isolation of faults to a hardware component, and recovery of operation after software reconfiguration so as not to use the fault-isolated component.

This operating system software also provided the system operator a means to change the system configuration. This manual control included the means to diagnose failed elements within a component, to repair the failed elements and to return the component to operational use, concurrently with maintaining real-time operation.

The real-time requirements for automatic casualty recovery were determined by a systems design and by detailed analysis of the requirements which supported mission success. The process of developing the specialized software was complex. This process required a detailed understanding of hardware operation

7.4 (continued)

and software requirements, careful preparation of design for both the hardware configuration and operating system software, accurate coding and debugging, and a long period of extensive testing. Extensive testing occurred over the period spent testing the integrated operational program (four years on Program 1, four years on Program 2, and two years on Program 3). This extensive testing was supported by many software tools including real-time simulation, data extraction/recording, and data reduction programs.

The degree of fault tolerance actually achieved in Program 1 is being measured while this report is being written. Early results show that a figure close to 60% of all actual failures are being automatically recovered, whereas the anticipated design objective was set at 40%.

SECTION 8
REAL-TIME ON-LINE MAINTENANCE TESTING (TASK 2.7)

8.0 INTRODUCTION

The real-time on-line maintenance testing consisted of various forms of testing programs incorporated into the real-time combat systems. Execution was initiated:

- a) Periodically, automatically in the passage of time.
- b) By operator console intervention.
- c) Upon error detection by the operating system.
- d) Upon subroutine call from a real-time operational system task.

The remainder of this section describes the following types of testing with supportive tabular information:

- . Confidence testing (table 8-1, 8-2).
- . Diagnostic testing in reduced mode.
- . System Interface testing (table 8-3).
- . System Operability tests (table 8-4, 8-5).

TABLE 6-1. PROGRAM 1 CONFIDENCE TESTING

Test No.	Test Identification	Function Description	Core Words Required	Execution Time	Frequency Execution	Mode of Execution	System Recovery Time
10.	COMMON SYSTEMS						
11.	CSMIOT	QUICK MAIN MEMORY ADDRESSING OR IOC CONTROL MEMORY TEST	203	240 USEC TO 1 MSEC	BEFORE EACH I/O REF * MEM TEST PRIOR TO MEM LOAD	USER CALLED AND BEFORE I/O TO LOAD, RELOAD	ALTERNATE IOC IS SWITCHED WITH ERR DETECTION 1.5 SEC TO 5 MIN.
12.	CSARITH	QUICK FLOATING AND FIXED POINT ARITHMETIC TEST	54	125 USEC	USER CALLED	USER CALLED SUBROUTINE	RECOVERY, STOP CPU
13.	CSCPU	A CPU CHECK ON THE ALTERNATE PROCESSOR	15	15 USEC	1 HZ	PERIODIC	STOP OTHER CPU
14.	ALTERNATE RECOVERY	PROCESS THE CLASS I INTERRUPTS IDENTIFYING MEMORY RELATED FAILURES				AUTO RESPONSE TO CLASS I INT.	2.5 SEC. DISK DATA TRANSFER RATE DEPENDENT
15.	NORMAL RECOVERY	PROCESS THE CLASS I, II, III HARDWARE ERROR INTERRUPTS	1127	120 MSEC MAX	UPON FAILURE DETECTION	AUTO RESPONSE TO CC INT.	0-1.5 SEC. FAILURE DEPENDENT
16.	MODULE CONFIG. RESPONSE TEST	IOC, CLOCK, MEMORY RESPONSE TESTS	40	20 USEC	USER CALLED AS REQUIRED	INITIATED WHEN CALLED	0. ELEMENT NOT RESPONDING IS SET DOWN
20.	CPU - PERIODIC CONFIDENCE TESTS	ALL PERIODIC CONFIDENCE TESTS ARE RUN IN ONE SERIAL PROGRAM UNTIL ERROR IS DETECTED	460	200 USEC/EACH	1 SEC	TASK & INT. TAKES CONTROL OF ALL CL II INTERRUPT PROCESSING	CPU STOPPED, PROCESSOR PLACED IN DOWN CONFIG. WHEN ERROR IS DETECTED
*21.	EXECUTIVE STATE REGISTER	TEST OPERABILITY OF INTERRUPT BASE, INDEX, ACCUMULATOR STORAGE PROTECTION, AND SEGMENT IDENTIFICATION REGISTERS. TESTS THE INITIAL CONDITION WORD AND DESIGNATOR STORAGE WORDS FOR EACH CLASS OF INTERRUPTS	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE
22.	MEMORY LOCKOUT	TEST ABILITY TO CONFINE TASK PROGRAM TO A DESIGNATED AREA OF MEMORY	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE
23.	CP MONITOR CLOCK	TEST ABILITY TO TRANSFER CPU CONTROL AT A PREDETERMINED TIME TO THE EXECUTIVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE
24.	PRIVILEGED INSTRUCTION	TEST ABILITY TO DETECT ATTEMPTED EXECUTIONS OF PRIVILEGED INSTRUCTIONS IN THE TASK STATE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE
25.	INDIRECT ADDRESSING	TEST ABILITY TO PERFORM INDIRECT ADDRESSING FUNCTIONS	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE
30.	IOC/ICA CONFIDENCE TEST						
31.	I/O CONTROL MEMORY	IOC CONTROL MEMORY TEST (OTHER THAN CSMIOT)	25	15 USEC	WHEN CALLED BY EXEC SERVICE REQUEST	SUBROUTINE OF EXECUTIVE	RECOVERY WITH REDUNDANT IOC

* NOTE: ITEMS 21 THRU 25 DESCRIBE THE SERIES OF TESTS PERFORMED WHEN THE PERIODIC CONFIDENCE TEST IS INITIATED INTO EXECUTION.

BEST AVAILABLE COPY

TABLE 8-2. PROGRAM 3 CONFIDENCE TESTS

Test No.	Test Identification	Function Description	Code Words Required	Execution Time	Frequency Suggested	Mode of Execution	System Recovery Time
10.	COMMON SYSTEM						
11.	CSMOT	QUICK MAIN MEMORY ADDRESSING IN 100 CONTROL MEMORY TEST	150	100 USEC TO 1 MSEC	BEFORE EACH I/O REF & MDI TEST PRIOR TO SYSTEM TO MEMORY	UPON USER DEMAND & PRIOR TO LOAD, RELOAD	1.5 SEC
12.	CSAMTN	QUICK FLOATING POINT AND FIXED POINT ARITHMETIC TEST	100	125-150 USEC	USER CALLED	USER CALLED SUB-ROUTINE	STOP CPU
13.	CSOPU	A CPU CHECK ON THE ALTERNATE PROCESSOR	50	100-110 USEC	PERIODIC, 100 MS.	PERIODICALLY AS EXEC STATE SUB-ROUTINE	STOP CPU. SEE REAL-TIME AUTO CASUALTY RECOVERY
14.	ALTERNATE RECOVERY	PROCESS THE CLASS I INTERRUPTS IDENTIFYING MEMORY RELATED FAILURES	100	2 MSEC	UPON FAILURE DETECTION	AUTO RESPONSE TO CLASS I INT.	1.5 SEC. DISK DATA TRANSFER RATE DEPENDENT
15.	NORMAL RECOVERY	PROCESS THE CLASS I, II, III HARDWARE ERROR INTERRUPTS	2000	125 MSEC MAX	UPON FAILURE DETECTION	INTERUPT RESPONSE	1.5 SEC. FAILURE DEPENDENT
16.	CPU CONFIDENCE TEST, (PERIODIC)	TEST NO. 15 IS A COM- PLETE CPU CONFIDENCE TEST. EXE- CUTED IN SERIAL FORM BEFORE FAILURE DETECTION	412	200 USEC	0.5 HZ	PERIODIC	1.5 SEC TO 5 MIN. SEE PROCES- SOR PLATED DOWN WHEN ERROR IS DETECTED
20.	EXECUTIVE STATE REGISTER	TEST OPERABILITY OF IN- TERRUPT BASE AND SEQUENCE IDENTIFICATION REGISTERS. TESTS THE CRITICAL CONDI- TION WORD AND DESIGNATOR STORAGE WORDS FOR EACH INTERRUPT CLASS	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE
22.	MEMORY LOCKOUT	TEST ABILITY TO CONFINE TASK PROGRAM TO A DESIGNATED MEMORY AREA	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE
23.	CPU MONITOR CLOCK	TESTS HARDWARE TRANSFER OF CPU CONTROL TO THE EXECUTIVE AT A PREDE- TERMINED TIME	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE
24.	PRIVILEGED INSTRUCTION	TESTS INTERRUPT PROCES- SING DETECTION OF APPROPRIATE EXECUTION OF PRIVILEGED INSTRUCTIONS IN THE TASK STATE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE
25.	INDIRECT ADDRESSING	TEST INDIRECT ADDRESS- ING FUNCTIONS	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE	PART OF 20 ABOVE
26.	ARITHMETIC TEST	TEST BASIC INSTRUCTION SET AND DESIGNATORS TO DETECTION OF AN ARITHMETIC PROBLEM. TEST USE OF TASK STATE CONTROL MEMORY	PART OF 20 ABOVE	PART OF 20 ABOVE	0.5 HZ	PERIODIC, DEFERRABLE	SEE 20 ABOVE
40.	100/10A CHANNEL TESTS	100/10A TEST VERIFYING OPERABILITY OF CHANNEL COMMUNICATIONS VIA BUFFERS	SEE 41, 42 BELOW	SEE 41, 42 BELOW	SEE 41, 42 BELOW	SEE 41, 42 BELOW	SEE 41, 42 BELOW
41.	INTERCOMPUTER A TO B	TIMED TEST OF 100 TO 100 OPERABILITY	75	140 USEC TO 1 MSEC (VARI- ANT TO SUP- PLEDED DATA WORDS)	2 HZ	PERIODIC TASK	2 SEC
42.	INTERCOMPUTER TO RADIO ROOM (A TO C) AND (B TO C)	TEST OF INTERCOMPUTER CHANNEL TO FROM AN- OTHER WITH TIMED TEST PATTERNS	100	VARIANT, TABULARIZED CASES	1 SEC	PERIODIC, 1 SEC	N/A
43.	PERIPHERAL DEVICES EXERCISED	DISK, MAG TAPE EXER- CISED	200	VARIANT, TABULARIZED CASES	AS USER CONSOLE REQUEST	CONSOLE REQUESTED	2 SEC
44.	END TO END SIGNAL DATA CONVERTER	END AROUND LOOP TEST WITH SIGNAL DATA CON- VERTER	100	DATA LENGTH DEPENDENT	3 SEC	PERIODIC	2 SEC

BEST AVAILABLE COPY

TABLE 8-3. PROGRAM 3 SYSTEM INTERFACE TEST

Test No.	Test Identification	Function Description	Core Words Required	Execution Time	Frequency Execution	Mode of Execution	System Recovery Time
10.	SC CONTROL MODULE SC CONTROL TESTS 1-19	CONTROL LOAD RELOAD OF TESTS, ELEMENTS, CPU SELECTION HARDWARE STATUS RECORD, TEST INITIATION, EXEC. START	2300 INST 2100 DATA	12 SEC 2 CPU TESTS	ALL LOADS, RELOAD	STATIC EXEC INDEPENDENT	5 MIN.
11.	CPU TO CPU INTERFACE	HARDWARE CPUA TO CPUB INTERFACE IN-TERESTY VERIFICATION	50	.002 SEC	ONCE AT LOAD OR RELOAD FOR EACH CPU	INITIATED BY SC	N/A
12.	MEMORY OPERAND BUS TEST	TESTS ALL MEM BANKS TO FROM OPERAND REFERENCE	60 INST	0.001 SEC	ONCE AT LOAD OR RELOAD FOR EACH CPU	INITIATED BY SC	N/A
13.	MEMORY INSTRUCTION BUS TEST	TESTS INSTRUCTION EXECUTION, CONFIRMATION IN CONTROL MEMORY	70 INST	.001 SEC	ONCE AT LOAD OR RELOAD FOR EACH CPU	INITIATED BY SC	N/A
14.	CPU I/O COMMUNICATION TEST	TEST CPU TO I/O, LOAD & TEST MONITOR CLOCK	20 INST	600 USEC	ONCE AT LOAD OR RELOAD FOR EACH CPU	INITIATED BY SC	N/A
15.	I/O CHAIN EXECUTION INSTRUCTION TEST	TEST I/O EXECUTION OF INSTRUCTION FOR EACH I/O TO MEMORY	40 INST	185 USEC/ I/O	ONCE AT LOAD OR RELOAD FOR EACH I/O	INITIATED BY SC	N/A
17.	COMPUTER MEMORY TEST	BIT PATTERN TEST CPU TOTAL MEMORY BANKS TEST	150 WORDS	.001 SEC	ONCE AT LOAD OR RELOAD PER I/O FOR 2 CPU	INITIATED BY SC	N/A
19.	PERIPHERAL TEST	TEST OF MASS STORAGE DEVICES FROM I/O INCLUDING DISK AND I/O CONSOLES	150 WORDS	PERIPHERAL DEVICE RESPONSE DEPENDENT .002 SEC	ONCE PER CPU I/O	INITIATED BY SC	N/A
20.	LOAD FAILURE ANALYSIS	PROVIDE CONTROL TO ALTERNATE RECOVERY FOR MEMORY RESUME INTERRUPTS TO A NON-RECOVERING MEMORY MODULE	NDRC ONLY	N/A	ONCE PER CPU LOAD/RELOAD START	MANUAL OR AUTO INITIATION CLASS 1 INTERRUPT	5 MIN.

NOTE: ALTERNATE RECOVERY PROVIDES SELECTION OF MMU'S, I/O, CPU CHANNELS, MASS MEMORY AND NOTIFIES OTHER COMPUTER CPU OF RELOAD. RELOADS SOFTWARE VIA NDRC BOOTSTRAPPING OF THE SOFTWARE CONFIGURATION (SC) MODULE.

TABLE 8-4. PROGRAM 1 OPERABILITY TESTS

Test No.	Test Identification	Function Description	Core Words Required	Execution Time	Frequency Execution	Mode of Execution
100.	ON-LINE POPA	PERIPHERAL POPA TEST OF I/O, CHANNEL, AND PERIPHERAL OPERABILITY	SEE 101, 102, 103, 104 BELOW	SEE 101, 102, 103, 104 BELOW	SEE 101, 102, 103, 104 BELOW	SEE 101, 102, 103, 104 BELOW
101.	MASS STORE DISK FILE	TESTS RANDOM ACCESS DISK CONTROL, DISK DRIVE CYCLINDER TRACK ADDRESS READ/WRITE	16,000 _B	3 HOURS	EACH OPERATOR CONSOLE REQUEST	REQUESTED TASK
102.	SIGNAL DATA CONVERTER	TESTS CONTROL AND DATA TRANSFER END AROUND (LOOP) WITH SIGNAL DATA CONVERTER	10,200 _B	1 HOUR	EACH OPERATOR CONSOLE REQUEST	REQUESTED TASK
103.	DATA EXCHANGE I/O CONSOLE	TESTS CONTROL AND DATA TRANSFER TO MAGNETIC TAPE MASS STORAGE AND CONTROL OF OTHER PERIPHERALS	16,500 _B	1 HOUR	EACH OPERATOR CONSOLE REQUEST	REQUESTED TASK
104.	SECOND I/O CONSOLE	TEST CONTROL AND DATA TRANSFER TO KEYBOARD PRINTER	3,500 _B	15 MIN.	EACH OPERATOR CONSOLE REQUEST	REQUESTED TASK

BEST AVAILABLE COPY

TABLE 8-5. PROGRAM 2 OPERATIONAL TESTING

Test No.	Test Identification	Function Description	Core Words Required	Execution Time	Frequency Execution	Mode of Execution
100.	ON LINE PCFAS	PERIPHERAL PCFA TEST OF 100, I/O CHANNEL PERIPHERAL OPERABILITY (PERIPHERALS ARE DEDICATED TO TEST, UNAVAILABLE TO REAL-TIME)	SEE 101 THROUGH 110 BELOW	SEE 101 THROUGH 110 BELOW	SEE 101 THROUGH 110 BELOW	SEE 101 THROUGH 110 BELOW
101.	MAINTENANCE CONTROL CONSOLE TEST	TESTS MAINTENANCE CONTROL CONSOLE, SYSTEM MAINTENANCE CONSOLE CONTROL AND DATA TRANSFER INTERACTION	8,024	2 HOURS FOR 1 EXECUTIONS	UPON OPERATOR COMMAND	UPON REQUEST BY TEST CONTROLLER
102.	DATA EXCHANGE I/O CONSOLE TEST	TESTS MAG TAPE AND KEYBOARD PRINTER I/O FUNCTIONS IN PRIMARY AND BACKUP CONSOLES, PAPER TAPE, I/O AND TELETYPE	7,168	2 HOURS FOR 15 EXECUTIONS	SAME AS 101 ABOVE	SAME AS 101 ABOVE
103.	SECOND I/O CONSOLE TEST	TESTS KEYBOARD PRINTER I/O FUNCTIONS	2,560	1 HOUR FOR 2 EXECUTIONS	SAME AS 101 ABOVE	SAME AS 101 ABOVE
104.	SITUATION SUMMARY DISPLAY CONSOLE TEST	TESTS KEYIN	13,312	1 HOUR/CONSOLE FOR 2 EXECUTIONS	SAME AS 101 ABOVE	SAME AS 101 ABOVE
105.	VIDEO SIGNAL SIMULATOR (VSS) TEST	TESTS CONTROL FUNCTIONS AND VIDEO GENERATION DATA OF VSS	12,288	3 HOURS FOR 2 EXECUTIONS	SAME AS 101 ABOVE	SAME AS 101 ABOVE
106.	RADAR VIDEO PROCESSOR TEST	TESTS CONTROL AND RADAR AZ-RANGE DATA CONVERSION	3,370	2 HOURS FOR 2 EXECUTIONS	SAME AS 101 ABOVE	SAME AS 101 ABOVE
107.	BEACON VIDEO PROCESSOR TEST	TESTS CONTROL AND BEACON RADAR DATA GENERATION	2,417	2 HOURS FOR 2 EXECUTIONS	SAME AS 101 ABOVE	SAME AS 101 ABOVE
108.	SHIP TO SHIP/SHORE COMMUNICATIONS TESTS	TESTS CONTROL AND DATA TEST MESSAGE BLOCK EXCHANGE WITH OTHER COMMUNICATION EQUIPMENT	4,608	2 HOURS FOR 2 EXECUTIONS	SAME AS 101 ABOVE	SAME AS 101 ABOVE
109.	AIR TO SHIP/SHORE COMMUNICATIONS TESTS	TESTS VTA SIMULATION TEST MESSAGE INPUT I/O OF AIR TO SHIP/SHORE	4,608	2 HOURS FOR 2 EXECUTIONS	SAME AS 101 ABOVE	SAME AS 101 ABOVE
110.	SIGNAL DATA CONVERTER TESTS	TESTS VTA END AROUND (ICUP) HARDWARE SIGNAL DATA CONVERTER	6,656	4 HOURS FOR 6 EXECUTIONS	SAME AS 101 ABOVE	SAME AS 101 ABOVE
111.	INTERCOMPUTER I/O CHANNEL	TESTS I/O CONTROL TEST MESSAGE PATTERN OF 32 BIT INTERCOMPUTER CHANNEL	10,980	1 HOUR FOR 12 EXECUTIONS	SAME AS 101 ABOVE	SAME AS 101 ABOVE
112.	MASS MEMORY DISC CONTROL & STORAGE TEST	TESTS CONTROL AND STORE, FETCH DATA ON MASS STORAGE TEST ALLOCATIONS AREAS	9,024	1 HOUR OR 3 HOURS WITH END FOR 5 EXECUTIONS	SAME AS 101 ABOVE	SAME AS 101 ABOVE
200.	OTHER SUBSYSTEM OPERABILITY TESTS	TESTS MAJOR SUBSYSTEMS PERIPHERAL HARDWARE PLACED IN TEST MODE	N/A	N/A	UPON OPERATOR INTERVENTION	INITIATED AS TASK AFTER LOAD FROM MASS STORE BY TEST LEADER UPON DETECTION OF CONSOLE INTERVENTION
201.	COMBAT SYSTEM ALIGNMENT TEST	TESTS AND REGISTERS CORRELATION CORRECTION FACTORS FOR RADAR DATA ALIGNMENT AND FIRE CONTROL MISSILE GUIDANCE TRACK	5,248	7 STAGES OF 1 TO 2 HOURS EACH	SAME AS 200 ABOVE	SAME AS 200 ABOVE
202.	COMBAT SYSTEM OPERATIONAL TESTS	TESTS RADAR SETS AND HARDWARE I/O INTERFACE INCLUDING RADAR DATA CORRELATION QUALITY	5,464	9 HOURS	SAME AS 200 ABOVE	SAME AS 200 ABOVE

BEST AVAILABLE COPY

8.1 CONFIDENCE TESTING DETAILED DESCRIPTIONS

Confidence testing was used to determine an accepted level of confidence of component operability if no failures were detected during its execution. The program was also run periodically or upon detection of failure for normal error processing to reaffirm a constant (hard) failure, or upon a need to verify a repaired or reinstated hardware component.

Confidence test programs were initiated through one of the following:

- a) A background periodic or deferred mode (after other priority real-time tasks).
- b) Other error analysis software upon detection of hardware error.
- c) User personnel by console intervention.
- d) Subroutine call from the application task program.

8.1.1 Program 1 Confidence Tests

Table 8-1 describes the various confidence tests involving Program 1 in a multiprocessor configuration system.

8.1.2 Program 2 Confidence Tests

No confidence tests, by definition, were provided for Program 2.

AN/UYK-7 diagnostics could be loaded in real-time in a reduced hardware capability mode for either diagnostic purposes or for confidence testing.

8.1.3 Program 3 Confidence Tests

8.1.3.1 General Control

The performance monitoring appraisal (PM) module provided control of the on-line C&CS hardware confidence test. It also con-

8.1.3.1 (continued)

sisted of the interface software for operator console insertion and control of the operators selected confidence tests. Table 8-2 describes the confidence tests provided in this project multiprocessor configuration system.

In table 8-2, the system recovery time portrays the time and event associated with returning to a real-time operation after error detection and recovery from a failed state condition.

8.1.3.2 Unique Implementation Features

The Navy controlled MODX2 Standard Executive provided the confidence tests for intercomputer component functions. Further changes to this furnished baseline MODX2 Standard Executive were implemented as Engineering Change Proposals (ECPs) to provide essential alterations. These alterations directed failure isolation and recovery from confidence test error detection. Confidence tests for computer A-to-B intercomputer were a part of the reconfiguration routines and intercomputer handler in Program 3 and initiated monitor tests of the other computer status, as well as I/O buffer timing. The MODX2 Standard Executive contained the intercomputer channel interface for the handler, and it monitored channel use by performing a two-word confidence test prior to each transfer of an intercomputer message buffer.

Both Computer A-to-B had a backup redundant channel on a separate AN/UYK-7 IOC. Intercomputer I/O was reestablished on these backup channels if either primary channel failed.

The confidence test for radio room intercomputer channel was a test of AN/UYK-7 to/from AN/UYK-20 intercomputer channel use. Several patterns were used in this test of the channel both ways. The C&CS operator was notified if the communications failed either way.

8.1.3.2 (continued)

Peripheral time-out tests were provided as an ECP for the re-design of the Sea Test Model of Common Peripheral (MODX2). The data converter module of Program 3 provided the periodic end-around (loop) test on the Signal Data Converter. This circumvented real-time operations interference. Again redundant SDCs and cross-coupled dual I/O channels provided back-up switch-over capability.

8.1.3.3 Tools and Methodology for Program 3 Confidence Tests Development

Program 3 provided confidence tests, fault isolation and recovery of fault for devices in the central computer complex (computers, intercomputer channels, and system shared peripherals). Program 3 included the PM (control) module, but no on-line peripheral Programmed Operational Functional Appraisal (POFA) or diagnostic tests.

The tools and methods of test development for Program 3 are described in other sections of this report entitled Top-Down Software Development, Software Development Standards, and Host Development Support Software.

8.2 DIAGNOSTICS TESTING IN REDUCED CONFIGURATION MODE

Diagnostic testing was conducted on failed elements after they were placed in an inactive operational state. The remaining active hardware elements provided a reduced capability operational system.

8.2.1 Program 1 Diagnostics

A Casualty Diagnostic Interface (CDI) routine of 400 words was loaded into the system memory upon operator console intervention in realtime requesting this CDI. Further interaction with the operator permitted the loading of 16,000 words of AN/UYK-7 GFE diagnostics. All the diagnostic segments were executable. A memory bank was tested in about one second and an arithmetic CPU or IOC element was tested in approximately one second. These tests provided display output to isolate failures to as low as the three card hardware level. These diagnostics did not perform multiprocessing tests. The diagnostic segments were executed serially on given elements and provided success/completion display when no error was detected. They were open ended to facilitate adding special complex tests for unique configuration requirements. The total test duration was dependent on configuration requirements. The total test duration was dependent on configuration complexity. A mean time to repair goal of 15 minutes dictated the need to exercise against diagnostic in approximately 5 minutes.

8.2.2 Program 2 Diagnostics

For Program 2, the AN/UYK-7 diagnostics were also loadable upon operator console intervention during real-time operation. This capability was similar to that provided in Program 1. Further variations existed in the actual diagnostics as dictated by the four mainframe on AN/UYK-7 configuration.

8.2.3 Program 3 Diagnostics

The CDI routine of Program 1 was replaced by use of the PM module in Program 3. This PM module was used to facilitate the operator interface for selectively loading and executing the Government Furnished Equipment (GFE) AN/UYK-7 diagnostics or other confidence or operability tests. The PM modules, as developed in Program 3, loaded the entire diagnostics from a peripheral mass memory into main memory. Other Program 3 elements (reconfiguration routines and software configuration) provided the control for reconfiguring hardware elements to "down" or unavailable status. This placement of an element to a "down" status established reconfigurations to degradable hardware modes.

For each computer, A or B, the components of one of the three mainframes could be placed "down", while the remaining components of two mainframes maintained a degraded C&CS operation. See section on Real-Time Automatic Casualty Recovery for further descriptions of this capability.

8.3 SYSTEM INTERFACE TESTING

The purpose of system interface testing ranged from operability confirmation of hardware intramodule interfaces to off-line diagnosis. These tests were generally operated as part of the system initialization and loading sequence.

The testing of hardware interfaces was performed to determine resources available to the combat operational system software and to insure hardware system integrity. POFA tests served to test the CPU, IOC, memory, I/O channel and peripheral interfaces. Other interface tests were performed on-line and during initialization and/or operation system load phase for Programs 2 and 3.

8.3.1 Program 1 System Interface Tests

Program 1 did not contain any unique tests for system interface testing.

The Combat System Interface Test (CSIT), comprised of program elements for test verification of all hardware elements interfacing into the system, was initiated from a separate CSIT load of the system prior to loading the operational system. The CSIT program subsystem contained more than 48,000 words of object code.

8.3.2 Program 2 System Interface Tests

The system interface test subsystem was executed as part of the System Loader (SL) operation. The hardware test control routine initiated a large comprehensive set of program elements to verify various hardware interfaces. Tests of interfaces between AN/UYK-7 components included:

8.3.2 (continued)

- a) Memory to CPU to memory interface
 - 1) Operand Bus Interface.
 - 2) Instruction Bus Interface.
- b) CPU to IOC Communication
- c) IOC to memory to IOC
 - 1) Operand Bus Interface.
 - 2) Instruction Bus Interface.
 - 3) Data Transfer Bus to I/O Channels Register (IOC - DRO Memory Communication).
- d) IOC Channel Test of all channels
 - 1) Load from memory or register to IOC control memory.
 - 2) Store from IOC control memory to main memory or general registers.
- e) Peripheral Tests

Determination of on-line, off-line status for the following peripherals by exercise of control via IOC to peripherals.

 - 1) I/O Console
 - 2) Mass Storage Devices
- f) Memory Bank Test
 - . Bit pattern storage and retrieval of all cells in each memory bank.

Each of the above tests were run in a serial manner. As each test was successful, a status flag marked availability for the elements tested. If a failure occurred in tests a, b or c, a HALT (4 STOP) condition resulted with a status value in index 5.

8.3.2 (continued)

In the case of tests e or f, failures resulted in marking the hardware elements as unavailable in status. Furthermore, loader functions used the status flags to locate or condition system software to the availability of hardware. The test control of SL performed tests a and b for each of two CPUs.

Approximately 2,800 instruction words comprised the test routines with 200 words in the test control routine. A successful execution of all tests was conducted within twelve seconds.

8.3.3 Program 3 System Interface Tests

The Program 3 system interface test was also performed at system initialization prior to real-time operation initiation or upon system recovery reload. The Program 3 Software Configuration (SC) Module was loaded and initiated by the system load initialization or upon recovery reload. One segment of this module performed system hardware interface integrity tests of the Central Computer Complex hardware. This series of tests could only be initiated upon successful bootstrap load or reinitialize load. The success to load partially tested the NDRO-CPU memory. After the first series of CPU hardware tests were executed, a dual processor CPUA-to-CPUB test was performed. When successful, CPUB interface tests were conducted, and after successful completion, the real-time operating system and resident control modules were loaded and initiated. See table 8-3.

8.4 System Operability Testing

System operability testing was initiated periodically in the background or was called from a task state module or user console. The channels and peripherals under operability testing were defined as unavailable to the real-time system during the test.

8.4.1 Program 1 Operability Testing

The operability tests, in general, were on-line POFAs. Table 8-4 describes the tests provided for Program 1.

8.4.2 Program 2 Operational Testing

Twelve POFA tests and two specialized subsystem operational tests constituted the operability testing for program 2 (see table 8-5).

8.4.3 Program 3 Operability Testing

Sperry Univac provided four programs in the system which test the operability of the I/O channels. These tests are already described in paragraph 8.1.3, Program 3 Confidence Tests. Sperry Univac did not supply the peripheral POFA tests on this C&CS system.

The four tests for C&CS system operability described in Table 8.1.3 were:

- 1) Signal Data Converter Tests
- 2) Disk Mass Storage Data Recorder Tests
- 3) Intercomputer A to Computer B I/O Tests
- 4) Intercomputer A to AN/UYK-20 (Radio Room Computer) Test

8.5 OVERVIEW ON TOOLS AND METHODS OF TEST PROGRAM DEVELOPMENT

Test programs on programs 1, 2, and 3 were developed primarily through use of a CMS-2 compiler on a Host Univac 1108/RIPS. Some of the early AN/UYK-7 hardware tests for the Program 1 project were also run on the AN/UYK-7 simulator hosted on the 1108. Master source and object files were maintained on the host 1108. Advantages of the use of the host software on the Univac 1108/RIPS were as follows:

- a) Rapid program generation capability.
- b) Some debugging via simulation (non I/O dependent).
- c) Minimizes development time frame.
- d) Operates on limited real equipment capability.
- e) Convenience and standard job control technique.
- f) Reduced computer time/costs.
- g) Improves reliability of software.
- h) Host software file utilities, editor simplified target system master file maintenance.

8.5.1 Top Down Test Software Development

The Program 3 test software was developed using the tools and methods described in the Top-Down Program Development section.

SECTION 9
HOST DEVELOPMENT SUPPORT SOFTWARE (TASK 2.8)

9.0 INTRODUCTION

This section describes and discusses developmental support software tools hosted on large-scale data processing systems. These tools were used as an aid in the development of real-time military systems. Where data was available, comparisons are made on cost effectiveness and manpower reduction factors as experienced by programs 2, 3, and 4.

The fact that Sperry Univac has used these tools on all of these projects over a span of years substantiate their value to the company. Research shows that each new project created new, or expanded the capabilities of existing software development tools. Continuous evolution of support tools has increased efficiency for developing software.

9.1 USAGE OF DEVELOPMENT SUPPORT SOFTWARE AND TOOLS

Table 9-1 shows the tools used on each of the four programs studied. The tools are cross referenced to the paragraphs describing them. The host computer is a UNIVAC 1100 series computer provided by a large commercial time-sharing facility.

TABLE 9-1. DEVELOPMENT SUPPORT SOFTWARE AND TOOLS USAGE

		Program 1	Program 2	Program 3	Program 4
ELEMENT	DESCRIPTIVE PARAGRAPH				
HOST MANAGEMENT TOOLS:					
1. CSTAR	9.2.5	X		X	
2. PROMIS	9.2.18				X
3. RIPS.MANPOWER	9.2.23	X	X	X	X
4. MRR	9.2.16	X	X	X	X
5. LIBRARY	9.2.13	X	X	X	X
6. RIPS.CORE	9.2.21	X	X	X	X
7. GRAPH	9.2.12	X	X	X	X
HOST PROGRAM DESIGN TOOLS:					
1. FLOWCHARTS	9.2.10	X	X	X	X
2. RIPS.CORE	9.2.21	X	X	X	X
3. RIPS.TIPO	9.2.31		X		
4. DOD	9.2.7			X	X
5. DYNAPROBE	9.2.8	X	X		
6. PET	9.2.17	X			
7. STRUCTURED NARRATIVE	9.2.32	X			X
HOST PROGRAM GENERATION TOOLS:					
1. ULTRA/32	9.2.33	X	X	X	
2. EXEC-8 TOOLS	9.2.9	X	X	X	X

TABLE 9-1. DEVELOPMENT SUPPORT SOFTWARE AND TOOL USAGE (cont.)

5. AUTOMATED TESTS	9.2.2	X	X	X	
6. RIPS.RSG	9.2.24	X	X	X	X
7. RIPS.TIPO	9.2.31		X		
8. RIPS.RUN	9.2.25	X	X	X	X
9. WRAP-AROUND SIMULATION	9.2.35	X	X	X	X
10. RIPS.COMPAR	9.2.20	X		X	X
11. RIPS.TAPED	9.2.28	X		X	X
12. PET	9.2.17	X			
13. DYNAPROBE	9.2.8	X	X		

HOST PROGRAM CONFIGURATION MANAGEMENT TOOLS:

1. RIPS.RSG	9.2.24	X	X	X	X
2. RIPS.RUN	9.2.25	X	X	X	X
3. RIPS.ED	9.2.22	X	X	X	X
4. RIPS.BLDSIT	9.2.19	X	X	X	X
5. RIPS.TAPIN	9.2.19	X	X	X	X
6. RIPS.TAPOUT	9.2.30	X	X	X	X
7. RIPS.SYSIN	9.2.26	X		X	X
8. RIPS.SUSPIT	9.2.27	X		X	X
9. RIPS.COMPAR	9.2.20	X		X	X
10. RIPS.TAPED	9.2.28	X		X	X
11. DOD	9.2.7			X	X
12. MAPPER	9.2.15	X	X	X	X
13. GRAPH	9.2.12	X	X	X	X
14. LIBRARY	9.2.13	X	X	X	X
15. LOG CMS-2 TAKS	9.2.14		X		

TABLE 9-1. DEVELOPMENT SUPPORT SOFTWARE AND TOOLS USAGE (cont.)

3. FORTRAN V	9.2.33	X	X		X
4. CMS-2Y	9.2.9	X	X	X	X
5. RIPS.RSG	9.2.24	X	X	X	X
6. RIPS.RUN	9.2.25	X	X	X	X
7. CMS-2 SYSMAKER	9.2.3		X		
8. RIPS.ED	9.2.22	X	X	X	X
9. RIPS.BLDSIT	9.2.19	X	X	X	X
10. RIPS.TAPIN	9.2.29	X	X	X	X
11. RIPS.TAPOUT	9.2.30	X	X	X	X
12. RIPS.SYSIN	9.2.26	X		X	X
13. RIPS.SYSOUT	9.2.27	X		X	X

HOST PROGRAM DOCUMENTATION TOOLS:

1. FLOWCHARTS	9.2.10	X	X	X	X
2. DOD	9.2.7			X	X
3. GRAPH	9.2.12	X	X	X	X
4. STRUCTURED NARRATIVE	9.2.32	X			X

HOST PROGRAM TEST TOOLS:

1. AN/UYK-7 SIMULATOR	9.2.10	X	X	X	X
2. UPAK	9.2.34	X		X	X
3. DEBUG AIDS	9.2.6	X	X	X	X
4. CMS-2Y	9.2.4	X	X	X	X

9.2 DESCRIPTIONS OF SUPPORT TOOLS

The following subparagraphs describe each of the support tools.

9.2.1 AN/UYK-7 Simulator

This support program, running on the UNIVAC 1108 computer, provided the capabilities of AN/UYK-7 program generation and preliminary program checkout on the UNIVAC 1108. The following functions are provided:

- a) The AN/UYK-7 instruction repertoire simulator.
- b) The CMS-2 compiler/loader simulator.
- c) The CMS-2 monitor simulator.
- d) The common program MODX2 simulator.
- e) Program checkout aids.

This simulator allowed mathematical and functional program checkout on the host computer. Once preliminary checkout was completed on the host computer, final checkout was completed on the AN/UYK-7 computer.

9.2.2 Automated Common Program Certification Test Support Software

A package of test support software was developed for the Common Program. This support software was the certification of the Common Program in Programs 1, 2, and 3. It consisted of the following:

- a) An automated test controller.
- b) Scenario input from punched cards, disk, paper tape, or magnetic tape (see figure 9-1).
- c) A data recording module to extract information during realtime operation and record it on magnetic tape; and

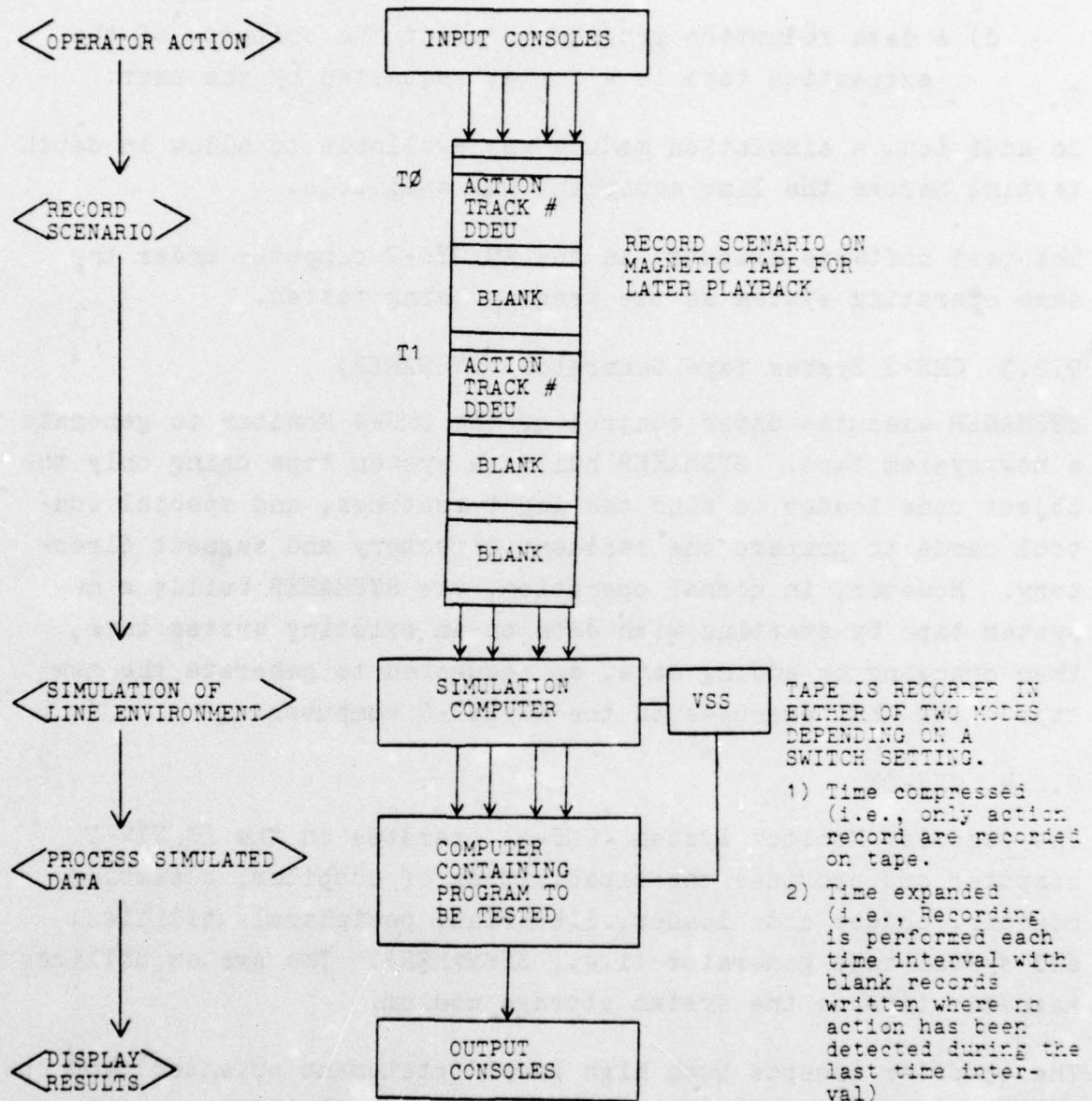


Figure 9-1. Simplified Schematic of Scenario Usage

9.2.2 (continued)

- d) A data reduction program to print the contents of the extraction tape in a format requested by the user.

In addition, a simulation module was available to allow in depth testing before the line equipment was available.

The test software executed in the AN/UYK-7 computer under the same operating system as the program being tested.

9.2.3 CMS-2 System Tape Generator (SYSMAKER)

SYSMAKER executes under control of the CMS-2 Monitor to generate a new system tape. SYSMAKER builds a system tape using only the object code loader to bind the input routines, and special control cards to prepare the resident directory and segment directory. However, in normal operation, the SYSMAKER builds a new system tape by starting with data on an existing system tape, then changing or adding data, as requested to generate the new tape. SYSMAKER executes on the AN/UYK-7 computer.

9.2.4 CMS-2Y

The Compiler Monitor System (CMS-2) operates on the AN/UYK-7 computer and provides the capabilities of compiler, assembler, monitor, object code loader, librarian, peripheral utilities, and system tape generator (i.e., SYSMAKER). The system utilizes magnetic tape as the system storage medium.

The compiler accepts both high level (statement oriented) and low level (computer instruction mnemonic oriented) languages. These languages describe the desired program. From these descriptions the compiler generates the object code data to be loaded into the computer memory as an executable program by the object code loader.

9.2.4 (continued)

The assembler differs from the compiler in that it accepts only the low level input language. The assembler allows the use of macros.

The monitor is a serial batch processing executive.

The loader performs instruction absolutizing of object code produced by the CMS-2 compiler and assembler.

The librarian provides a means of storing, retrieving, and updating source statements and relocatable object code. It is capable of updating either entire elements or individual items within an element.

The peripheral utilities provide a variety of functions for manipulating data files on the peripheral devices. These functions include:

- a) Position a specified magnetic tape to the start of a file.
- b) Position a specified magnetic tape to the start of a record within a file.
- c) Read tape into memory.
- d) Write tape from memory.
- e) Compare the contents of two magnetic tapes and print out any differences.

The system tape generator (SYSMAKER) provides a method of updating system tapes which have a directory scheme the same as that used for the CMS-2 system tapes. The system tape generator accepts input that contains the data to change the tape directory information and then using the object code loader, generates a new system tape complete with required directories.

9.2.5 Cost/Schedule Technical Achievement Reporting System (CSTAR)

This support program runs on the host computer and is an integrated program management system complies with Department of Defense Instruction 7000.2, AFSCP 173-3, NAVMAT P5240, and AMCP 3705.

CSTAR provides a formal means for planning, scheduling, budgeting, measuring progress, reporting, and controlling a program or a proposal. CSTAR uses a basic approach of definition, schedule, budget, and uses budget dollars as a standard unit of measurement to establish progress. This process is followed for all levels of program effort such as projects, organizations, Work Breakdown Structure elements, cost accounts or work packages.

CSTAR consists of procedures, charts, documents, and a group of computer programs which not only integrate the planning data, budget tables, rate and factor tables, with actual costs, but also generates computerized output reports.

9.2.6 Program Management Information System (PROMIS)

This support program which runs on the host computer, provides Program Management with a formal planning, budgeting, progress measuring, reporting, and controlling capability regarding a specific program. PROMIS uses the Work Breakdown Structure (WBS) as a tool and concept in maintaining product identification while transmitting information and acquiring data. WBS is the same concept used in the larger and more complex CSTAR system. In fact, PROMIS can be considered a major subset of CSTAR which gives many of CSTAR's advantages. It provides sufficient visibility to monitor program progress in scheduling and cost, while reducing materially the high costs of implementation and maintenance inherent in the larger system.

9.2.6 (continued)

Like CSTAR, PROMIS consists of procedures, charts, a group of computer programs which integrate the planning data, budget tables, etc. with actual costs and generates computerized output reports.

9.2.7 Design Object Drawing (DOD)

This support program, which runs on the host computer, provides the capability to automatically produce design object drawings (or any other "tree structured" type of diagram which can be represented within the limits of the database). Output from this program can be either inked drawings using the XYNETICS[®] plotter, or microfilm using the UNIVAC 1617 COM (computer output microfilm). A tabular summary of the drawing data is optional. Input data consists of a file of special control cards.

DOD has been a valuable tool for generating the hierarchical tree functional flow diagrams used to depict the developing system. Its input may easily be generated manually during initial design, or automatically directly from the program code during maintenance for documentation updates, or for verification that the code matches the design.

9.2.8 Dynaprobe

Dynaprobe is a hardware device used to measure system utilization. It measures either the time duration, or frequency of an event within the hardware. Probes are used to detect the event. A system monitor process records the data on magnetic tape. Dynapar data reduction software running on the Host Computer reduces the data and produces the system utilization reports. While designed primarily for measuring hardware efficiency by probing applicable memory locations, it can also measure software usage.

[®] XYNETICS is a registered trademark of Xynetics Electroglass Incorporated

Examples of Count Mode measurements include the number of seeks on a random access file, the number of "switches" between the Supervisor and a given partition in core, and the arrival rate of messages at the line controller or CPU.

A simplification of the measurement process is illustrated in Figure 1-2. Monitoring the host computer system involves the following steps:

- . Probe Attachment
- . Monitor Operations
- . Monitor Data Recording
- . Data Reduction

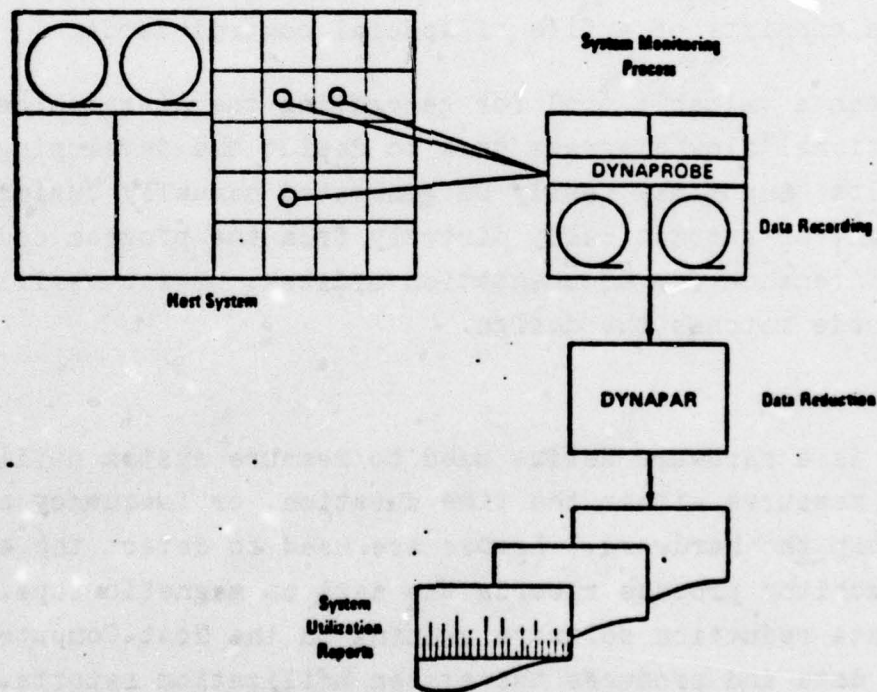


Figure 9-2. Dynaprobe-7900/DYNAPAR Functional Flow

9.2.9 EXEC-8 Tools

The host computer software support tools provide extensive text editing and file handling capabilities to ease the task of CMS-2 program updating and maintenance. When used with the RIPS system, the extensive time-sharing capabilities allow concurrency of CMS-2 source program preparation and updating, generation of CMS-2 runstreams and listings, and production of CMS-2 tapes for transference to standard configuration sites.

These tools are available 24 hours a day and execute in either demand or batch processing modes. In addition, the high speed random access capabilities of the 1100 mass storage subsystems increase the system throughput of CMS-2 operations in the AN/UYK-7 computer.

For a complete list of the different EXEC-8 tools available, see the Sperry UNIVAC 1100 series Executive System Programmer Reference Manual.

9.2.10 Flow Charts

This support program, which runs on the host computer, allows the user to automatically generate the low-level functional flow diagrams (flow charts) by processing any source program deck containing flow chart directives. These directives are contained in the comment areas of the source deck and take the form of ".X" where X represents one of the available flow charting options. This sequence is usually followed by text to be included in the flow charts.

Flow charts generated by this program satisfy the requirements of the American National Standards Institute and military specification WS-8506.

9.2.11 FORTRAN V

This support program, which runs on the host computer, provides a compilation capability into the 1108 instruction language. Sperry Univac has had considerable experience in software trans- portability and FORTRAN is the most used tool for transferring 1100 software to software capable of executing correctly on other computers.

9.2.12 GRAPH

This support program, which runs on the host computer, provides the user with the capability to generate a large variety of graphs quickly and economically on the XYNETICS plotter or on the UNIVAC 1617 COM (computer output microfilm). The program processes a deck of cards, or input file of card images, con- taining the commands and graph data and generates a XYNETICS or COM driver tape. The XYNETICS plotter, or the COM, reads the tape and generates the graphs.

One of the chief uses of GRAPH to date has been for the manpower loader charts (see RIPS manpower) where conversion programs are available to automatically generate the GRAPH input. However, graph is applicable to a long list of uses and is expected to be used widely on future projects.

9.2.13 LIBRARY

A central depository is maintained for Programs 1, 3, and 4 for all documentation generated or acquired by these projects re- lating to their work. This filing system is maintained by a librarian who is responsible for signing out the documents and maintaining indexes and cross references. A similar system used by Program 2 without a librarian showed a significant loss of control.

9.2.13 (continued)

The following are advantages of the library system:

- a) Limits the reproduction costs involved with furnishing multiple copies.
- b) Makes all documents more accessible by maintaining them at a central site.
- c) Reduces the storage space required to provide individual copies of many documents.

9.2.14 Log CMS-2 Tasks

As a configuration management tool, Program 2 included, at the end of each 1108 runstream, a series of EXEC-8 commands which made a permanent record of all CMS-2 jobs completed. A history file was established and each runstream created a dummy element in this file. This element consisted only of a descriptive name and version number, but this was sufficient to establish a record of modifications to each program. Since the file was never packed, printing the table of contents resulted in a chronological record of execution and maintenance runtime for each program which included task name/version, date and time of each event.

9.2.15 MAPPER

This support program, which runs on the host computer, provides a report processing capability. The user may list his reports in a variety of formats. It has been used for many different applications (e.g., STR reporting). The system is capable of reporting entry, storage, retrieval, realtime update/change, and hardcopy output. Searches, sorts, and computations may be performed and output. The user may generate his own report format for his particular application.

9.2.16 Management Responsibility Reporting (MRR)

This support program, which runs on the host computer, collects, maintains, and reports direct costs that are incurred on efforts. These direct costs are listed by individual responsibility.

These periods management reports are generated for:

- a) Program Level.
- b) Project Level.
- c) Control Task Level.
- d) Control Task Summary.
- e) Project Detail Task.

The different level reports are valuable to individuals responsible for costs at the program, project, and control task levels respectively.

Data for generating the reports is available from a direct cost master file of current and cumulative costs. Current open commitments and authorized budget figures are also available for reference.

9.2.17 Processing Evaluation Tool (PET)

This support program, which runs on the AN/UYK-7 computer, is used to extract timing information for program during real-time operational conditions. PET is a task state program easily transferable to other systems utilizing the common program executive operating system. PET operates in either one or two processors. PET creates a magnetic tape recording for each program entry and exit.

9.2.17 (continued)

Two types of reports are generated either on-line, or off-line from the tape created by PET (see figures 9-3 through 9-5):

- 1) Percentage of processor time per module for each entry. It also breaks this down into the time spent in each CPU.
- 2) Record of the maximum amount of time spent in one entrance and the total number of times that entrance occurred within the requested time period.

When reducing the data contained on the extraction tape, the operator may specify the time period he desires reduced. Only data from recordings made during this period will be considered for the report.

9.2.18 RIPS.BLDSIT

This support program, which runs on the host computer, reads binary card images from a RIPS* standard input tape and writes them to a temporary card input file on a drum. This card input file is then available for execution.

9.2.19 RIPS.COMPAR

This support program, which runs on the host computer, provides the user the capability to compare two RIPS format drum files and produce a printed log of the differences. For example, it allows the user to determine all patches added to a baseline system. The file format and printer listing are options specified to contain a log of all the differences between the two input files.

* - Information concerning RIPS is Company Proprietary to Sperry Univac.

Program 1 Sample Timing Data

TIME 1447

		% Time in Each Entrance						
		PRI	PER	MES	DEF	DEM	I/O	I/E
CPU0	12.011%	4.492%	16.308%	2.441%	3.027%	0.292%	61.425%	Load With Multi Activities
CPU1	20.703%	1.953%	10.351%	0.000%	2.343%	0.097%	64.550%	
		PRI	PER	MES	DEF	DEM	I/O	I/E
CPU0	16.503%	4.492%	11.035%	3.125%	2.441%	0.390%	62.011%	Load by Large Single Activity
CPU2	12.011%	5.078%	3.393%	0.000%	2.636%	0.195%	71.679%	
TIME 1435		PRI	PER	MES	DEF	DEM	I/O	I/E
CPU0	5.468%	2.441%	10.742%	2.343%	3.710%	0.390%	74.902%	Average
CPU1	10.156%	2.148%	9.667%	1.464%	1.757%	0.097%	74.707%	Case
TIME 1650								

CPU - Central Processor Unit
 PRI - Priority entrance
 PER - Periodic entrance
 MES - Message entrance
 DEF - Deferred entrance
 DEM - Demand entrance
 I/O - Input/Output entrance
 I/E - Idle plus Executive time

Figure 9-3. Sample Timing Data

PROGRAM 1 Sample Timing Data

PET *T MOD 0012.35 0012.40/
MOD : 0012 35 0012.40

	<u>PRI</u>	<u>MES</u>	<u>PER</u>	<u>DEF</u>	<u>DEM</u>	<u>I/O</u>	<u>TOTAL</u>
FLM CP 0	3.85%	0.00%	0.00%	0.83%	2.44%	0.00%	7.1200%
CP 1	3.51%	0.00%	0.00%	1.17%	2.24%	0.00%	6.9800%
FPR CP 0	0.00%	0.00%	1.70%	0.00%	0.00%	0.00%	1.7000%
CP 1	0.00%	0.00%	3.02%	0.00%	0.00%	0.00%	3.0200%
CS CP 0	2.44%	0.14%	2.14%	0.00%	0.48%	0.73%	6.0000%
CP 1	4.49%	0.24%	1.66%	0.00%	0.34%	0.48%	7.2700%
FDM CP 0	0.00%	0.00%	0.19%	0.00%	0.00%	0.00%	0.1900%
CP 1	0.00%	0.00%	0.24%	0.00%	0.00%	0.00%	0.2400%

Figure 9-4. Sample Timing Data

PROGRAM 1 Sample Timing Data

PET *T MENT 0012.35 0012.40/
MENT 0012.35 0012.40

	<u>PRI</u>		<u>MES</u>		<u>PER</u>		<u>DEF</u>		<u>DEM</u>		<u>I/O</u>	
	<u>NE</u>	<u>MT</u>	<u>NE</u>	<u>MT</u>	<u>NE</u>	<u>MT</u>	<u>NE</u>	<u>MT</u>	<u>NE</u>	<u>MT</u>	<u>NE</u>	<u>MT</u>
FLM CP 0	+117	+4	+0	+0	+0	+0	+2	+23	+123	+1	+0	+0
FLM CP 1	+115	+4	+0	+0	+0	+0	+2	+21	+108	+1	+0	+0
FPK CP 0	+0	+0	+0	+0	+35	+5	+0	+0	+0	+0	+0	+0
FPK CP 1	+0	+0	+0	+0	+59	+5	+0	+0	+0	+0	+0	+0
CS CP 0	+31	+10	+21	+1	+106	+3	+0	+0	+26	+1	+60	+1
CS CP 1	+52	+11	+26	+3	+24	+3	+0	+0	+21	+1	+34	+1
FOM CP 0	+0	+0	+0	+0	+122	+1	+0	+0	+0	+0	+0	+0
FOM CP 1	+0	+0	+0	+0	+113	+1	+0	+0	+0	+0	+0	+0

Figure 9-5. Sample Timing Data

9.2.20 RIPS.CORE

This support program, which runs on the host computer, provides the user with the capability to generate a report listing the current size, in decimal, of the number of computer words now required by each of the projects program modules. This provides a total program size determination with a high degree of accuracy and makes it a valuable program management tool.

The information is obtained from the 1108 RIPS format files containing the programs object code. The user provides the file name and listing format desired.

9.2.21 RIPS.ED

This support program, which runs on the host computer, updates CMS-2 source libraries from a 64 column uniscope terminal. It has the following advantages over using \$LIBEXEC:

- a) Many users may simultaneously perform CMS-2 source library updates.
- b) Procedures may be randomly accessed for updating.
- c) Minor source card images changes may be made by minor manipulation rather than repunching entire statements.
- d) CMS-2 source library elements are resequenced for easy reference from the CMS-2 compiler listings.

9.2.22 RIPS.MANPOWER

This support program, which runs on the host computer, gave the user the capability to generate periodically current and projected manpower loading (backlog) reports. The reports could be obtained from a terminal display, as a printer listing, or in graphic form (see figure 9-6) via a conversion routine (WAVBLOG) and the GRAPH program. Input to the system is generated along

9.2.22 (continued)

with each proposal and is updated as needed for each current contract. A sample input form is shown in figure 9-7. Reports could be requested for all projects, or for a specific cost center.

Manpower projections were listed by categories, where category I represents manpower needs for contracts currently being worked on. Category II represents manpower needs for projects funded internally to Sperry Univac. Category III represents projections of anticipated contract requirements each reduced by the estimated probability that Sperry Univac will receive this contract.

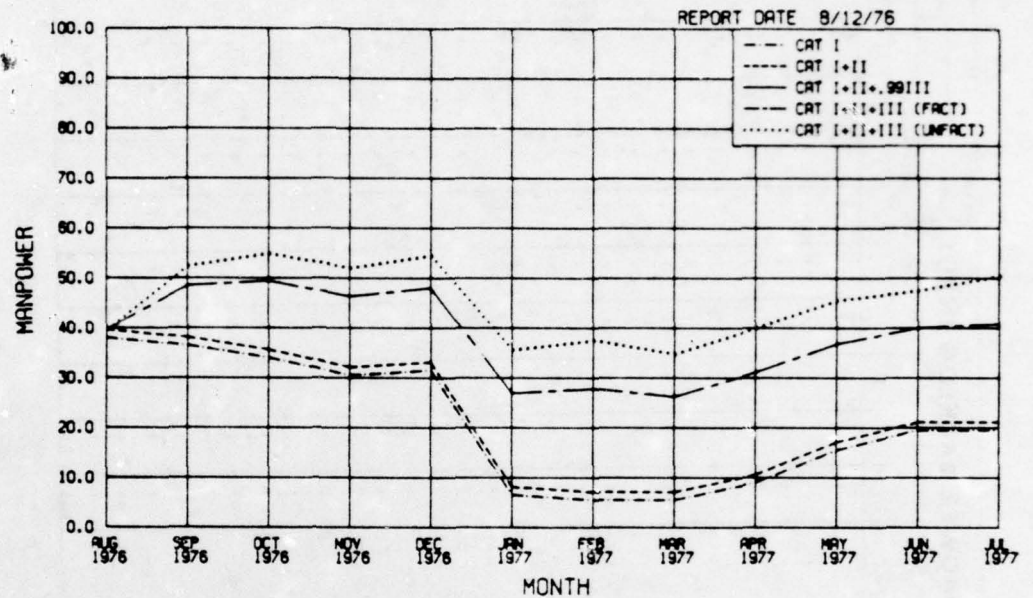
The reports show the expected manpower loading as it appears at the time of generating a report. Therefore, these reports vary dynamically from week to week depending on the status of pending contracts.

RIPS.MANPOWER furnishes an effective management tool.

9.2.23 RUNSTREAM GENERATOR (RIPS.RSG)

This support program, which runs on the host computer, provides the user with an easy means of creating and controlling a large variety of Exec 8 runstreams from a runstream skeleton and a user specified set of substitution parameters. It is an extremely useful tool for projects which have a large number of (very similar) runstreams to create, maintain, and control. The user specifies the processor operation and listing control options, the element containing the runstream skeleton to be processed, and one to nine character strings to replace designated strings in the skeleton.

MANPOWER BACKLOG COST CENTER



I	38.0	36.5	34.0	30.5	31.5	6.5	5.5	5.5	9.0	15.5	19.5	19.5
I-II	39.6	38.1	35.6	32.1	33.1	8.1	7.1	7.1	10.6	17.1	21.1	21.1
I-II-III (FACT)	39.6	48.5	49.4	46.3	48.0	26.9	27.8	26.2	31.1	36.8	40.0	40.8
I-II-III (UNFACT)	39.6	52.4	54.9	52.0	54.5	35.6	37.5	34.8	40.0	45.5	47.5	50.4

Figure 9-6. Manpower Backlog

9.2.24 RIPS.RUN

This support program, which runs on the host computer, executes the user's job under the CMS-2 Monitor in the AN/UYK-7 computer. This command is transmitted after creation of a runstream using RIPS.BLDSIT.

9.2.25 RIPS.SYSIN

This support program, which runs on the host computer, allows the user to read a CMS-2 system tape and convert it to a bootstrap format file. The bootstrap format is required by the other RIPS program maintenance processors. The user specifies the printer listing, tape formats, tape and file names, and the mapping desired from system format to bootstrap format.

9.2.26 RIPS.SYSOUT

This support program, which runs on the host computer, allows the user to create a user defined system tape. The output of a CMS-2 load operation is a bootstrap format file, either created directly by the loader (i.e., an LOBJECT file) or written by the utility processor (via BOOTWRT commands). The SYSOUT processor allows the user to produce from this bootstrap file, a system tape in a user specified format. It is a useful tool for allowing users to maintain their systems on the host in a bootstrap format file and automatically produce a system tape whenever one is needed for testing or delivery.

A mapping of the desired tape format, tape control, file names, and printer listing control information are specified by the user.

9.2.27 RIPS.TAPED

This support program, which runs on the host computer, provides the user the capability to patch or update a bootstrap format drum file. This operation frequently precedes a conversion from

9.2.27 (continued)

bootstrap format to system tape format. The bootstrap format file serves as a baseline against which patches can be applied and against which updated files can be compared. Available editing capabilities are as follows:

- IAC - Inspect and change
- SCH - Search for Data
- STR - Store Data
- DIS - Display Data
- X - Exit
- H - Help.

The operator at his option could specify a listing, applicable files, desired listing format, desired memory address, and size.

9.2.28 RIPS.TAPIN

This support program, which runs on the host computer, allows a user to input a magnetic tape file into a RIPS-format drum file. The format of the input file, tape control, and printer listing are options specified by the user. Input file formats may be either CMS-2 system tape format, or bootstrap format.

9.2.29 RIPS.TAPOUT

This support program, which runs on the host computer, allows a user to produce a magnetic tape file from a RIPS format drum file.¹ The format of the output file, tape control, and printer listing options are specified by the user.

9.2.30 Trace Input Process Output (RIPS.TIPO)

This support program, which runs on the host computer, allowed the user to trace the effect of an input stimulus (e.g., a con-

¹ A RIPS format file is a simulated magnetic tape file.

9.2.30 (continued)

sole button press) through the various program modules and interfaces affected by following the associated intermodule message traffic.

Data for the RIPS format TIPO file was manually generated and the file constructed with an element for each program module. This data for each module consisted of input message identifiers, names of processes performed, output message identifiers, triggers, and dummy output and external output designators.

The TIPO program begins with the element corresponding to a particular external stimulus. The first input message entry is designated a "trigger" entry and has the same name as the element in which it resides. TIPO lists the trigger element name followed by the name of each process used by the associated program module. When all processes have been listed, TIPO follows each output message name to its linked element and repeats the listing of processes contained in the element. Each branch of the chain is terminated by an external output. When all chains have been processed the program terminates. Dummy elements refer to loops encountered such as request/acknowledge exchanges. These are required to avoid a looping situation in the program.

TIPO provides the user with a visual trouble shooting tool for analysis of large complex programs. Currently data for the TIPO file must be generated and maintained by manual procedures. Automation of these are necessary to achieve maximum use of this tool. For example, on Program 2, analysis of the TIPO output listing aided in redesigning the message traffic and interface logic to cause them to operate more efficiently and thus materially reduce the processor load.

9.2.31 Structured Narrative

Structured narrative is a technique used in software documentation whereby program design flow is described by a sequence of English language formal constructs. The allowable constructs are listed in table 9-2. The result (see figures 9-8 through 9-10 and table 9-3 of a structured narrative with its corresponding Design Object Drawing, Design Object Summary, and Program Listing) is a clear, concise specification of a programs design.

9.2.32 ULTRA 32 Macro Assembly System

The Macro Assembly System for the AN/UYK-7 computer consists of an assembler, loader, librarian, and utilities. Each component represents an individual program which is interfaced with a system control program and a centralized input/output program. The assembler system of programs operates on the AN/UYK-7, generates object code for the AN/UYK-7, and supports the programmers/operators with loading, librarian, and utility capabilities.

9.2.33 Utility Package (UPAK)

The Utility Package (UPAK) is a stand-alone program designed for non-real-time use. It provides the programmer with the software necessary to load, verify, change, and dump specified areas of core storage. UPAK was developed as part of the common program Mod X2 executive operating system, but is actually a separate entity with its own executive and operates in complete independence of the common program. UPAK runs on the AN/UYK-7 computer and is used to build common program system tapes and system disks, to patch these systems during debugging, and to take dumps after common program execution.

FORMAL CONSTRUCTS	EXAMPLES	NOTES
<u><if:phrase></u> IF <condition> THEN <sentence:list> ELSE <sentence:list> ENDIF	IF this is true THEN Do this ELSE Do this instead ENDIF	
<u><case:phrase></u> CASE <condition variable> OF <label> :<sen- tence:list> <label> :<sen- tence:list> : : ELSECASE <sentence:list> ENDCASE	CASE returned status OF OK: establish bid for count eof: CALL cleanup GOTO ter- minate ELSECASE set error status ENDCASE	if OK stat do this if eof status do all of this
<u><goto:phrase></u> GOTO <name>	GOTO 1 GOTO scheduler	The name may be a label or the name of another narrative
<u><goto:at:phrase></u> GOTO <narrative:name> AT <label>	GOTO scheduler AT entry 1	Allows entering a narrative at a labelled sentence (use this sparingly)
<u><call:phrase></u> CALL <narrative:name>	CALL Magtape hand- ler	Used to refer to a narrative which will be implemented as a procedure or function

TABLE 9-2. FORMAL CONSTRUCTS

FORMAL CONSTRUCTS	EXAMPLES	NOTES
<u><loop:phrase></u> label:LOOP <sentence:list> ENDLOOP	main:LOOP Pick up interrupt status IF interrupt occurred THEN GOTO pro- cessing ENDIF ENDLOOP	LOOP-ENDLOOP brackets an in- finite loop The label is optional
<u><exit:phrase></u> EXIT <label list>	main:LOOP s1 s2 submain:LOOP s1 s2 EXIT sub- main s3 s4 ENDLOOP s5 s6 ENDLOOP	transfers eye and mind to s5 can also be used with FOR
<u><read:phrase></u> READ <I/O description>	READ next control card	for all input type I/O
<u><write:phrase></u> WRITE <I/O description>	WRITE new dms record	for all output type I/O
<u><return:phrase></u> RETURN <information>	RETURN to calling routine RETURN with bad status	exit from pro- cedure or func- tion narrative

TABLE 9-2. FORMAL CONSTRUCTS (continued)

PROCEDURE ICBPPRI

```
IF Monitor Interrupt Code equal 0(16)
  THEN BEGIN
    Clear Timeout Check Flag (ICBLSTAT (0,TOCHECK))
    If Timeout Message Sent Flag (ICBLSTAT (0,TOMSGSNT)) equal 1
      THEN BEGIN
        Call I/O Data Transfer Resumption Processor (ICPBIORE)
      END

    If Reset Mode (ICBLNTMI (0,RESET)) equal 1
      THEN BEGIN
        Call Reset Mode Processor (ICBPRSET)
      END

    If Hold Mode (ICBLNTMI (0,HOLD)) equal 1
      THEN BEGIN
        Call Hold Mode Processor (ICBPHOLD)
      END

    If Operate Mode (ICBLNTMI (0,OPER)) equal 1
      THEN BEGIN
        Call Operate Mode Processor (ICBPOPER)
      END

    END

  Return Control to Standard Executive
END-PROC ICBPPRI
```

Figure 9-8. Example of Structured Narrative

ELEMENT 004 ENR AC 3 LOC	IC03H56 FUNCTION 5 LOC	AC 2 LABEL SYS-PROC.	SOURCE STATEMENT	MEMORIC LISTING	09/11/76	PAGE 44 CID 510 CR
0 0 00173	541000	0 00095 0	IC0PPR1	PROCEDURE IC0PPR1 SET ARI0, CTO00009-3, 00. 30	PRIORITY ENTRANCE	00SIC 336
.....	IC0PPR1	JEP	6/25/76	SIC 337
.....	PROGRAMMER JEP	SPERRY-UNIVAC	6/25/76	SIC 340
.....	FUNCTION	THIS PROCEDURE IS THE ICB PRIORITY ENTRANCE.		SIC 341
.....	LANGUAGE	CMS-2		SIC 342
.....	INPUTS	MONITOR INTERRUPT CODE		SIC 343
.....	OUTPUTS	N/A		SIC 344
.....	RETURNS	RETURN CONTROL TO SE		SIC 345
.....	SCHEDULING CALLS TO PROCEDURE IC0PPR1, IC0PPR2, IC0PHOLD, AND IC0POPER			ICB 346 SIC 347

Figure 9-9. Program Listing Corresponding To The Example Structured Narrative

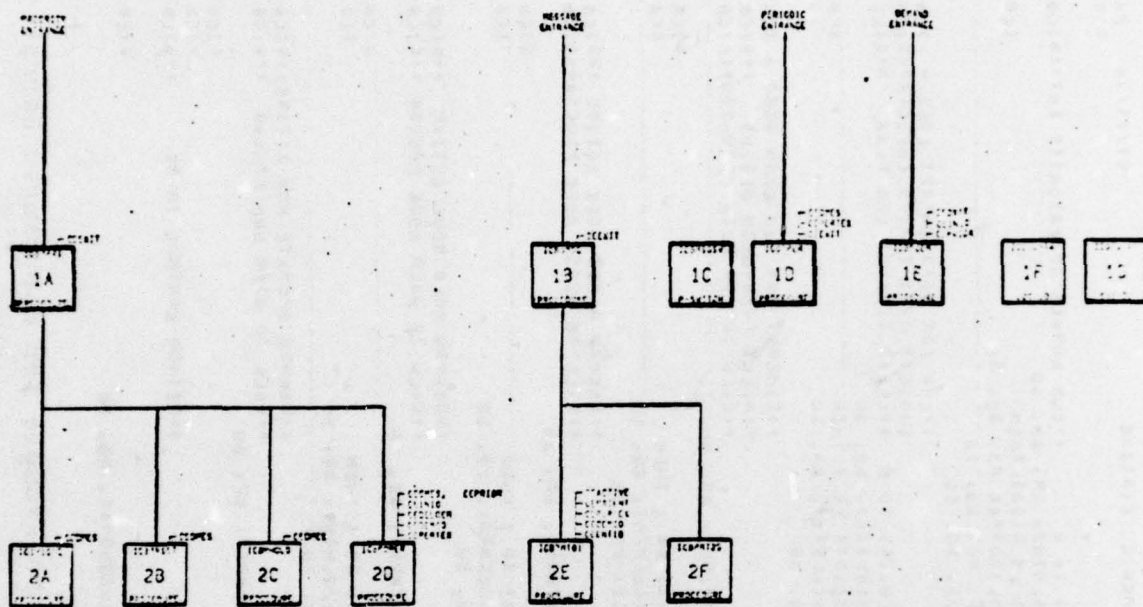


Figure 9-10. Design Object Drawing

D.O. ID	LANGUAGE ELEMENT	D.O. DESCRIPTION	CALLS TO FROM		ESRS	EXTERNAL CALLS	CPDS DESCRIPTION PARAGRAPH
1A	ICBPPRI PROCEDURE	PRIORITY ENTRANCE	2A 2B 2C 2D		CEEXIT		4.2.1
1B	ICBJMSG PROCEDURE	MESSAGE ENTRANCE	2E 2F		CEEXIT		4.2.2
1C	ICBMSGK P-SWITCH	MESSAGE TYPE P-SWITCH					4.2.3
1D	ICBPPER PROCEDURE	PERIODIC ENTRANCE			CEDMES CEPERTEX CEEXIT		4.2.4
1E	ICBPEM PROCEDURE	DEMAND ENTRANCE			CEEXIT CEINIO CEPRIOR		4.2.5
1F	ICBLDATA LOC-DD	ICB MODULE DATA STORES					4.2.6
1G	ICB7BRAT SYN-DD	BASE REGISTER ALLOCATION TABLE					4.2.7
2A	ICBPICRE PROCEDURE	I/O DATA TRANSFER RESUMPTION PROCESSOR		1A	CEDMES		4.2.8
2B	ICBPRSET PROCEDURE	RESET MODE PROCESSOR		1A	CEDMES		4.2.9
2C	ICBP HOLD PROCEDURE	HOLD MODE PROCESSOR		1A	CEDMES		4.2.10
2D	ICBPRER PROCEDURE	OPERATE MODE PROCESSOR		1A	CEDMES CEINIO CEDELOEM CEDEMIO CEPERTEX CEPRIOR		4.2.11
2E	ICBPYIC PROCEDURE	PRESET MESSAGE PROCESSOR		1B	CEACTIVE CEPRIENT CEPERTEX CEDEMIO CEENTIO		4.2.12
2F	ICBPMT25 PROCEDURE	SDC REGISTRATION MESSAGE PROCESSOR		1B			4.2.13

TABLE 9-3. DESIGN OBJECT SUMMARY

9.2.34 Debug Aids

The Debug Module of the common program provides general purpose realtime tools used to perform program debugging and system testing during integration. It is designed to have a minimal effect on the operating system. Therefore, it can be utilized in an operational system for debugging, modification, or system analysis. The Debug Module executes on the AN/UYK-7 computer.

Accessed by operator selections on the keyboard/printer, the Debug Module provides the following capabilities:

- a) INIT - Initializes the Debug Module for debugging with absolute or relative addressing.
- b) HIST - Gives an elapsed time between the execution of two specified breakpoints.
- c) TIME - Gives for two specified breakpoint addresses the average, standard deviation, and maximum and minimum execution times for N successive passes, plus consecutive individual execution times.
- d) CYCL - Gives a ratio of time spent between two breakpoints to elapsed time between each entrance into the breakpointed area.
- e) CHNG - Changes specified words in memory.
- f) SNAP - Presents the contents of selected memory locations and registers at a specified breakpoint.
- g) IMSG - Presents the specified intermodule messages during program execution.
- h) MODH - Presents the module history from the Preamble for a specified module.
- i) DUMP - Presents immediately the contents of specified portions of core storage.

9.2.34 (continued)

- j) SCON - Searches for a selected memory area for a specified masked constant.
- k) SOPR - Retrieves the addresses of certain instructions in a selected module that reference a specified operand.
- l) MGTR - Sends a specified intermodule message to a specified module.
- m) ABS - Retrieves the absolute address of a selected module segment.
- n) RPT - Repeats a debug request.
- o) SEQ - Repeats a debug request while incrementing a specified base address.
- p) CLR - Clears an uncompleted debug function.

9.2.35 Wrap-Around Simulation

Wrap around simulation is the technique of furnishing sufficient simulation software to simulate the entire real world environment experienced by the program under test. This allows the program to be tested and integrated to a high level of confidence long before the "live" equipment becomes available.

9.3 SOFTWARE DEVELOPMENT TOOLS EFFECTIVENESS

9.3.1 Core Reports (RIPS.CORE)

Program 2 was required to generate reports on the number of words of computer memory needed for each program module and the total required. At first this was generated manually from the program loader maps. When the RIPS support tools became available on the host computer the reports were generated automatically using RIPS.CORE. This program also converted the core sizes to decimal as an added convenience. Use of RIPS.CORE eliminated the 2.5 man hours per week formerly required to generate this report. The computer time required to search the program object code for the sizes was very small.

9.3.2 CMS-2 Monitor

The monitor was used to execute and debug certain special application programs, such as the mathematical equations used in the simulation function formulas. It could not be used for debugging the operational common program operating system.

For Programs 3 and 4 the monitor was combined with the RIPS (Remote Integrated Programming System). An interactive terminal operator could execute the program to be tested, collect data (SNAP, dumps, and trace data). After analysis of the collected data, the operator could make some desired software changes and rerun the tests.

Program 2 used the CMS-2 monitor in a stand-alone batch processing system. There was a 24 hour turn around delay, but this was still more cost effective than the "brute force" method of debugging at a dedicated computer system.

9.3.2 (continued)

Because of the complexity of the C&CS programs and problems with the dynamic hardware configurations at the dedicated site, testing costs were reduced by as much as 30% where the CMS-2 monitor was used.

9.3.3 Flowcharts

On program 2 flow charts were initially generated manually. However, because of the size and complexity of Program 2, flow charting directives were inserted in the comment fields of all new program source code to allow the flowcharts to be generated automatically via the host computer tool FLOWCHARTS.

Two advantages of using the automated FLOWCHARTS are as follows:

- 1) The flow charting directives provide sufficient comments to aid the understanding and readability of the program listing.
- 2) Updates to documented flowcharts consist of modification as required in the program code and execution of FLOWCHARTS to generate the flow chart update pages.

See Structured Narrative for a preferred method for describing program functional flow.

9.3.4 Functional Flow Diagrams

Manually generated high level functional flow diagrams were prepared for the Program Design Specifications for Program 1 and 2. In Programs 3 and 4 Design Object Drawings (DOD) were used for the functional flow diagrams. Initially the DODs were manually generated, but the final versions were generated on the UNIVAC

9.3.4 (continued)

1617 COM. An estimated cost saving of approximately 70% was experienced generating the flow diagrams via DOD as compared to the cost of generating them manually. In addition, the input files for DOD are easily updated and the job rerun to generate change pages for documentation changes due to program modifications.

9.3.5 Host Tools

The 1108 EXEC-8 operating system provided a complete set of software tools ranging from high-level language compilers to basic service functions. These included FORTRAN V, text editing and generation capabilities, file handling and I/O utilities. Once the Remote Integrated Programming System (RIPS) was operable, all of these tools were made available to the CMS-2 programmer via interactive terminals in the programming area. Thus, part of the reason Programs 3 and 4 were generated in a comparatively short time span was due to the availability of the host tools from their beginning.

9.3.6 Log CMS-2 Tasks

Log CMS-2 tasks were used on Program 2 to automatically maintain a history of program compilations, activations, etc. for configuration management purposes to retain control of software changes. Since the information was recorded automatically it ensured retention of the relevant information. Printed listings of the recorded information was readily available.

9.3.7 Program Compilation

Use of the 1108 RIPS for all CMS-2 compiler system needs has two major advantages over using a dedicated AN/UYK-7 CMS-2 facility:

9.3.7 (continued)

- 1) AN/UYK-7 compilation usage was reduced by approximately 44% when using RIPS (e.g., an average Program 2 module requires 45 minutes on a dedicated AN/UYK-7 computer and 25 minutes using RIPS).
- 2) Computer operator requirements were reduced significantly when using the time sharing 1108 RIPS interface.

9.3.8 Program Management Tools

CSTAR was used by Programs 1 and 3 because it was contractually required.

The PROMIS system was utilized on Program 4. PROMIS uses the WBS (like CSTAR) as its major organization guideline and gives many of CSTAR's advantages without the high costs of implementation and maintenance inherent in the CSTAR system. This resulted in an estimated paper work reduction of between 40% and 50%.

9.3.9 RIPS.TIPO

This tool was developed and used by Program 2. Since Program 2 was a very large complex program, some method was required to condense the information to manageable proportions for analysis of intermodule message traffic and responses to external stimuli. Trace Input Process Output (TIPO) satisfied this requirement and was very useful. However, it suffered from the drawback of having to be manually generated and maintained. Automating the generation of TIPO input data elements from the program code is possible future enhancement.

TIPO satisfies analytical needs on large programs. If functional flow sequence documents were to be required for a contract, TIPO with its sequential flow listings, would give multiple benefits and should be considered as a viable alternative

9.3.9 (continued)

to other forms of flow diagrams. Current estimates for production of TIPO elements average approximately one man week per major program module. Automation, of course, would reduce this figure drastically.

9.3.10 Runstreams

Runstreams, consisting of a sequence of system commands, eliminate the errors which are often encountered when attempting to execute a long sequence of complicated system commands. Once a runstream has been debugged, it may be activated by a single simple control statement and will then carry out the desired operations.

The runstream generator (RIPS.RSG) simplifies the generation of runstreams by creating a skeleton runstream. It tailors the skeleton to the users needs by inserting parameters furnished by the user. Where the user has many similar runstreams, RIPS.RSG provides the advantage of being able to make a common modification to all runstreams by just modifying the basic skeleton.

Use of runstreams to control batch jobs allowed for delayed execution to move computer loads to after peak hours. Runstreams were used by all programs studied and were especially valuable when combined with RIPS tools.

9.3.11 Scenario Controlled Testing

For the programs studied scenario controlled testing went through an evolution process. On Program 1 the initial input medium was punched paper tape. Because of the problem inherent with generating and reading large reels of paper tape the scenario were later maintained on magnetic disk.

9.3.11 (continued)

Program 2 went to a method using interactive display consoles to generate a magnetic tape for future play back. This "scripted" scenario tool provided dynamic generation of the desired tests. However, care must be taken in generating the tape to avoid inadvertent illegal key presses or other operator actions.

Programs 3 and 4 also recorded their tests on magnetic tape, but used a card deck input as a method for scenario generation control.

9.3.12 Structured Narrative

Structured Narratives were implemented in documentation during the later phases of Program 1 and Program 4. Because of its relative ease of generation and consequent cost advantages it is used as a desirable alternative to the flowcharts normally required by military specification WS-8506.

Structured Narrative is easy to generate because there is almost a one-to-one correspondence between it and the high level code used to generate the program. Thus, either the Structured Narrative is generated first as a design tool which is simple to convert to code, or the code is generated first and the documentation "falls out" of it.

The indented nature of the Structured Narrative graphically illuminates the relationships of the various program segments, while its condensed nature allows each page of Structured Narrative to replace (on the average) three or four pages of flow charts.

9.3.12 (continued)

It is not possible to measure the manpower and cost savings attributable strictly to the use of Structured Narratives. However, a comparative study of the documentation generated for two program modules, one documented using Structured Narrative and the other functional flow diagrams indicated a cost reduction of over 10% by employing Structured Narratives rather than flow diagrams in the descriptions. The general consensus of programming managers and supervisors polled on this subject shows them in agreement on the advantages of using Structures Narratives for documentation.

9.4.13 System Generation

Program 2 used CMS-2 SYSMAKER to generate system magnetic tapes for the AN/UYK-7 computer directly on the AN/UYK-7 computer. This has the advantage of allowing system tapes to be automatically regenerated.

On Programs 1, 3, and 4 the users maintained their systems on the 1108 Host Computer in a bootstrap format file and automatically produced a system tape whenever one was needed for testing or delivery. The users had the various EXEC-8 and RIPS tools available to expedite these operations. In particular, the RIPS SYSIN and RIPS.SYSOUT were used for system tape read and generation respectively. For Programs 1, 3, and 4, users of RIPS.SYSOUT estimate that system build time is reduced by approximately 8 hours over alternative methods used previously.

APPENDIX A
AN/UYK-7 COMPUTER OVERVIEW

INTRODUCTION

This appendix highlights several of the significant features of the Sperry Univac AN/UYK-7 computer. The information set forth in the following paragraphs is digested from the AN/UYK-7 TECHNICAL DESCRIPTION, Sperry Univac.

Central Processor Unit

The central processor unit (CPU) contains all the control, arithmetic, and timing circuitry required for instruction execution. Central processors operate in two different modes: the interrupt state executes the executive type functions, and the task state processes the application programs. A separate set of seven index, eight base, and eight arithmetic accumulator registers is available to the processor in each state. One to three central processor units may be configured in a totally shared memory system.

INSTRUCTION SET

The AN/UYK-7 is a 32-bit computer that offers 130 basic whole and halfword instructions that operate with 32-bit parallel, one's-complement, binary arithmetic. It contains both fixed and floating point hardware and can operate on 8-, 16-, 32-, or 64-bit operands. To complement its interrupt state mode, the AN/UYK-7 supports a set of privileged instructions which include input or output transfer initiation, read and control of the monitor clock, and control of the various processing activities in both the task and interrupt states.

AN/UYK-7 INSTRUCTION FORMATS

Bit No.	31	26	25	23	22	21	20	19	17	16	15	13	12	0
FORMAT I	f			a			k		b	i	s		y	
FORMAT II	f			a			f ₂		b	i	s		y	
FORMAT III	f			a			f ₃	k	b	i	s		y	

Elements of the Word are Interpreted as Follows:

Bit No.	31	26	25	23	22	20	19	17	16
Bit No.	15	10	9	7	6	4	3	1	0
FORMAT IV A	f			a		f ₄	b	i	
FORMAT IV B	f			a		m			

field	basic definition
f	6-bit function code
f ₂	3-bit subfunction code
f ₃	2-bit subfunction code
f ₄	3-bit subfunction code
a	3-bit accumulator register designator
k	operand interpretation designator
m	6-bit shift count designator
b	3-bit index register designator
i	indirect addressing designator
s	3-bit base designator

Instruction execution times vary from 1.0 microseconds to greater than 17.0 microseconds. The average instruction execution time is approximately 2.0 microseconds.

INSTRUCTION REPEAT

A novel feature of the AN/UYK-7 processor is its ability to repeat an instruction. The Repeat instruction causes any of a class of UYK-7 instructions to be repeated a user specified number of times. At every repetition, the count in a specified index register is reduced until zero is reached; for example, multiple loads, stores moves, and compares can be accomplished in short form.

ADDRESSING HARDWARE

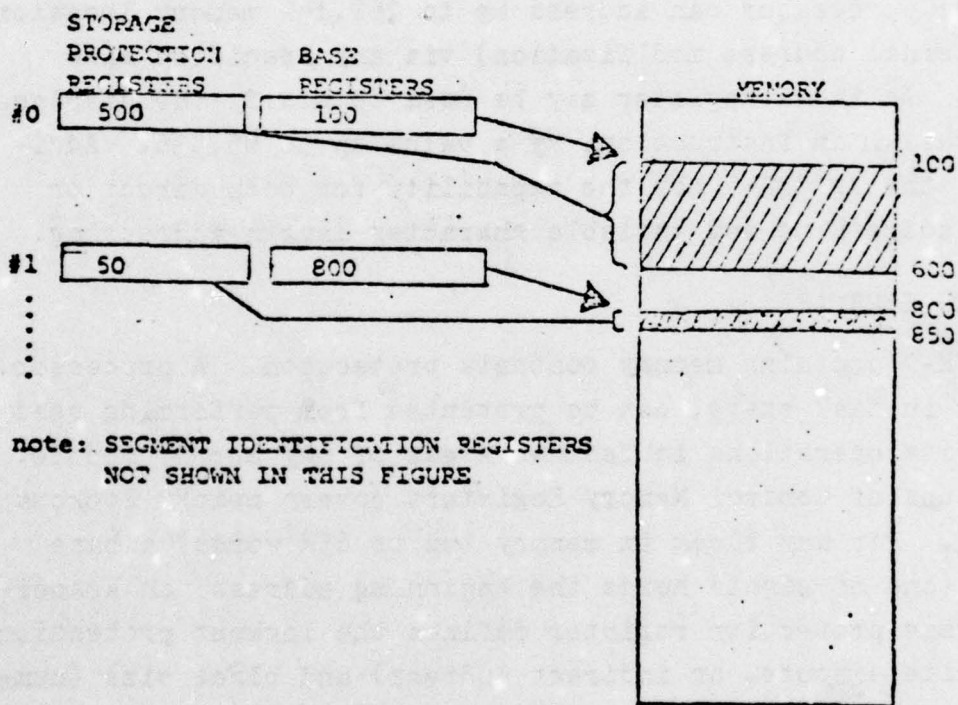
The central processor can address up to 262,144 memory locations (with internal address modification) via any specified base register. An index register may be used to modify the displacement address of an instruction, by a value up to 65,536. Additionally, the AN/UYK-7 has the capability for both direct or indirect addressing and variable character length addressing.

PROTECTION HARDWARE

The AN/UYK-7 contains memory contents protection. A processor, operating in task state, can be prevented from performing read and/or write operations in defined areas of any memory module. Three groups of Control Memory Registers govern memory lockout functions. For any block in memory (up to 65K words) a base register (one of eight) holds the beginning address; an associated storage protection register defines the lockout protection (read, write execute, or indirect address) and block size (number of words); and a segment identification register contains the relative address of the segment identity (that address in main memory from which the lockout information is transferred). Since all task state memory references utilize these base registers (and thus their corresponding storage protection registers), proper initialization of task state base and storage protection registers via the privileged instructions to access them allows the executive control of the application program memory

ADDRESSING HARDWARE (continued)

access. Since initiation of I/O is a privileged function. The executive program can provide any required degree of memory protection to the operation of IOC Command Chains and I/O buffers.



AN/UYK-7 HARDWARE PROTECTION EXAMPLE

INTERRUPTS

The interrupt mechanism of the AN/UYK-7 is fairly conventional. The interrupt classes are: hardware fault interrupts, CPU program faults and breakpoint interrupts, input and output program faults, and planned interrupts and program-initiated entrance to the interrupt state. When honoring an interrupt, the processor stores the current state of the machine and picks up appropriate new state information containing the starting location of the interrupt processing routine along with new status information such as lockout requirements for future interrupts. Interrupts that are currently locked out by the processor are held until such lockout is removed. Upon completion of interrupt return restores the state that existed at the moment of interrupt. This mechanism was dedicated CPU control memory registers for vectoring execution and saving former conditions upon interruption which are addressable to privileged instructions.

TIMING MECHANISMS

16-bit control memory register can be activated by software to decrease automatically its count to zero at the rate of 1024 counts per second and interrupt the CPU. Each IOC provide a separate oscillator that controls a 32-bit control memory register incremented at a fixed rate and a synchronized 16-bit control memory register that can be activated by software to decrement its count past zero and interrupt any accessible CPU. Additional IOCs clocks can be driven by an external oscillator.

MEMORY

Main memory is composed of modules of random access core storage with a read/write restore cycle of 1.5 microseconds. Each central processor or input/output controller can address up to 262,144 words (i.e., 16 modules of 16384 words each). Each processor also contains a 512 word NDR0 memory containing various

MEMORY (cont.)

firmware routines (e.g., interrupt analysis, initial load).

Recently Sperry Univac has optionally made available a replacement memory module called Double Density Mated Film Memory (DDMFM). DDMFM doubles the capacity of a memory module to 32,768 words and reduces the read restore cycle to 1.1 microseconds and the write restore cycle to 1.45 microseconds.

INPUT/OUTPUT

The UYK-7 controlled (IOC) contains the necessary control and timing circuitry to conduct orderly input and output transfers of data, external commands and external interrupts between accessible memory modules and the external devices on 4, 8, 12, or 16 full-duplex channels. IOC functions are governed by a chain of commands (set up by the central processor, but sorted in computer memory) initiated by one or more controlling central processors. I/O programs define buffer areas, channel numbers, and any functions related to word or byte size, imposed monitors, and transfer types. A processor has the capability to support up to four IOCs; a particular controller can be controlled by 1, 2, or 3 CPUs.

APPENDIX B

Overview of the Software Guidelines for Users of the Program 1 Common Program.

This document defined the system software structure and design rules required for use of the Program 1 Common Program by the tactical application programs for Program 1. The basic rules presented were:

- a) User programs operate in the AN/UYK-7 processor task state.
- b) User programs consist of program modules.
- c) User programs utilize the prescribed Common Program interface.

In addition to explicitly defining these rules, this document presented sections covering topics as follows:

- 1) Common Program functional description.
- 2) Use of the AN/UYK-7 Computer.
- 3) Predicting real-time performance of tactical function processing.
- 4) Compatible operation of the tactical application program modules.
- 5) Compatible operation of the tactical application program with the Common Program.
- 6) Guidelines for organizing and constructing a program module.
- 7) Glossary of abbreviations and terms used in this document.

CONCLUSION

The rules and guidelines presented in this document were a part of the requirements for the Program 1 application programs and were only allowed to be superseded by the formally controlled Program 1 System Specification. The operational success of Program 1 is attributed in part to these documented requirements.

APPENDIX C

OVERVIEW OF PROGRAM 2 MAINTENANCE INVESTIGATION ENGINEERING REPORT

INTRODUCTION

The Program 2 Maintenance Investigation Engineering Report presents the results of a study performed to evaluate the requirements of Program 2 advantageous to program maintainability. This report consisted of a set of general programming guidelines for use when designing and developing Program 2 with the CMS-2 Compiler system for use when designing and developing Program 2 with the CMS-2 Compiler system for use with the AN/UYK-7 Computer. The study was performed during the period from 1 July 1971 to 30 June 1972.

The investigation did not recommend changing the conventional method of program development, but provided detailed guidelines for program development to minimize problems of program maintenance. A summary of the individual areas addressed is as follows:

- 1) Program Documentation - this investigation defined guidelines to provide a orderly manner of organizing the documentation as defined in the NAVMAT Program Documentation Standard to coincide with the Navy maintenance agency philosophies and provide an ease of traceability.
- 2) I/O Formatters - this investigation addressed the I/O interface requirements relative to the systems data base; thus, the primary function was to format the system data into a data format required by the peripheral equipment/computer subsystem and vice versa.

INTRODUCTION (continued)

- 3) Program Structure - this investigation defined segmentation of the program to minimize duplication of data sets, techniques for accessing and modifying the system data base, and the structure of segments to enhance dynamic system reconfiguration, CMS-2 library control techniques, and maintenance of program listings.
- 4) Compiling Techniques - this investigation provided guidelines for CMS-2 Compiler system implementation, without causing cumbersome loader control, library control, and system tape building techniques. It provided guidelines for compiling, updating, creating a system tape, and automated patching of the program using the CMS-2 Compiler system. The guidelines provided were within the capability of Navy maintenance agency CMS-2 equipment suite.
- 5) Programming Practices - this investigation provided a programmer's reference manual that addressed labeling conventions, card deck ID conventions, comments, Common Program interfaces, and guidelines for program segmentation that are compatible with library control and CMS-2 provided program hierarchy.

The detailed analysis of this investigation resulted in the generation of programming guidelines, one part for each of the above listed areas and a report summary. An overview of the guidelines for each area is presented in the following paragraphs.

NAVMAT OPERATIONAL PROGRAM DOCUMENTATION GUIDELINES

This report presented detailed guidelines for implementing the NAVMAT standard, which was undergoing formulation and review by the Navy concurrently with the Program 2 development. This

NAVMAT OPERATIONAL PROGRAM DOCUMENTATION GUIDELINES (continued)

initial report contained detailed guidelines for the performance and design specifications. The final revised report was issued one year later and presented the detailed requirements for generation of the Program 2 documentation. This final document presented recommended requirements to four stages of program documentation; preprogramming documents, maintenance documents, test and acceptance documents, and operational documents. Specifically, this report recommended the following documents be produced:

Pre-Programming Documents:

Planning Documents:

- Tactical Operational Requirements
- System Design Data Document with Development Guidelines
- Interface Design Specifications
- Intrasystem Requirements Document

Design Documents:

- Performance Specification
- Design Specification

Maintenance Documents:

- Program Description
- Data Base Document
- Program Tape and Listing

Test and Acceptance Documents:

- Test Plan
- Test Procedures
- Test Report

Operational Documents:

- Operating Procedures Manual
- Command Manual
- Systems Maintenance Manual

NAVMAT OPERATIONAL PROGRAM DOCUMENTATION GUIDELINES (continued)

This report provided very definitive guidelines for the Performance Specifications, Design Specifications, Program Description Document and Data Base Document, and provided an overview to the NAVMAT Planning Documents, Test and Acceptance Documents, and Operational Document.

CONCLUSION

These documentation guidelines not only provided the standard for Program 2 documentation, but also influences the Navy definition of the NAVMAT documentation standard. In retrospect, the resulting performance and design specifications have proven effective direction to the analyst/programmer in the areas of design quality and software maintainability.

INPUT/OUTPUT FORMATTERS FOR PROGRAM 2

This report defined methods of implementing the use of Input/Output (I/O) formatters as applied to the peripheral equipment and computer sub-systems within the Command and Control System. The intent was that the I/O formatters contain buffers for each peripheral equipment/computer sub-system and convert information in common stores into a format and definition external to the internal requirements of the command and control system as required by the external device and vice versa. Thus, any modification to an external device did not affect the internal requirements of the command and control system other than the conversion requirements of the I/O formatter.

The approach taken during this study was to review the input/output requirements of Command and Control system to determine the feasibility of implementing the concept of input/output formatters and to define specific guidelines for selected peripherals. It was not the intent of this paper to define design requirements. Some areas of this paper did contain detailed design type information. This was necessary to describe an

INPUT/OUTPUT FORMATTERS FOR PROGRAM 2

approach to interfacing with an I/O formatter and was not to be construed as design requirements.

The intent of guidelines for the I/O formatters was to:

- . Minimize the impact of peripheral interface changes between peripheral input/output format requirements and the internal program requirements. Thus, the internal program design should not change unless additional capabilities were required.
- . Provide the capability for data link message structures to change without affecting the program design, provided that each data element meaning and range of values were unchanged within the central program.
- . Minimize the interdependency of tasks to:
 - a) conserve central program core space.
 - b) minimize timing problems
 - c) permit faster processing.
- . Provide standardization of message formats.
- . Provide standardization of display formats.

CONCLUSION

The guidelines for I/O Formatters were given Navy approval and were followed throughout the Program 2 design, production, and maintenance. This concept did succeed according to the stated intent and has proven to be of great value to program maintenance by structuring data interfaces and transfer.

PROGRAM STRUCTURE

This report documents a set of guidelines for creating the structure of Program 2. These guidelines provided standardized programming practices for design, production, and maintenance of Program 2. These guidelines were based upon the AN/UYK-7 interface requirements, the Common Program interface requirements, the capabilities of the CMS-2 compiler, librarian and loader, and the requirements of the C&CS operational program.

The approach taken during this study was to examine structures of NTDS programs to determine what changes in program development and methodology could be implemented to improve program maintenance. This resulting report provides guidelines for instructing Program 2 with respect to:

- . task allocation.
- . data item allocation.
- . common data access.
- . dynamically changing vehicular track capacities.
- . base register allocation.
- . executive interface requirements.
- . common routine re-entrancy considerations.
- . system configuration.

This report did not attempt to design the program, nor allocate tasks or data items, but provided guidelines for performing these functions. This report contained five primary technical discussions: Task Processing; Data Base; Common Program Interface Requirements; Common System Logic Structure; and Dynamic System Reconfiguration Requirements.

CONCLUSION

The Program Structure recommended for Program 2 was expanded in detail as parts titled Compiling Techniques and Programming Guidelines. These detailed guidelines were directions to the software analysts and programmers, and were enforced before the program code was accepted for incorporation into the developmental operational program. The resulting software code was of uniform structure such that maintenance, compilation and system tape generation control streams were identical in format for any program module. This has made Program 2 efficiently maintainable as a system program down to the level of program module and below.

COMPILING TECHNIQUES

This report defined guidelines for use of CMS-2 to create user application program system tapes. These guidelines were based upon binding user application program with the Common Program. The specifics stated in this study applied directly to the Program 2 system, and were intended to simplify and standardize application program generation and to enhance the program maintainability.

The approach taken in this study was to examine the techniques implemented for NTDS using CS-1, the techniques in use by AN/UYK-7 CMS-2 applications and requirements for the NTDS CMS-2 maintenance agency. The intent was to define use of the CMS-2 tools to create an operational system tape by providing a simple man/machine interface. The areas addressed in this study were CMS-2 Compilation, Library Control, Loader Control, and System Tape Generation. These guidelines extended the guidelines given in Program Structure and capitalized on the modular structure for application programs using the Common Program and the tools provided by CMS-2 to create guidelines for compiling any program module using minimal man/machine interfaces.

COMPILING TECHNIQUES

This study recommended creation of a system tape generator to be a tool for any program module by meeting the following requirements:

- . Generation of a tape directory to satisfy Dynamic System Reconfiguration requirements.
- . First block on tape to be written in bootstrap format as required by the AN/UYK-7 hardware bootstrap as currently designed for the Program 2 computer complex.
- . Capability to add/delete/replace a module, or a segment of a module and update the directory.
- . Capability to add patches during a system tape generation and update the tape directory.

The system tape builder (SYSMAKER) that had been developed for creating CMS-2 System tape was found as the most compatible available tool for these requirements.

CONCLUSION

The system tape builder for Program 2 was the existing SYSMAKER together with minor modifications and added as a system component to the CMS-2 system. The resulting modified SYSMAKER thus became a tool available from the Navy Controlled CMS-2 system for system tape generation for AN/UYK-7 programs. The guidelines for this report have been since implemented as series of automated tools provided on the Univac 1108/HOSTED CMS-2 system.

PROGRAMMING GUIDELINES

This report presented guidelines for using the CMS-2 to create application program modules of Program 2, and therefore, interface with the Common Program. These guidelines defined uniform programming practices and development techniques for application program modules. These guidelines supplemented the CMS-2 User

PROGRAMMING GUIDELINES

Reference Manual; thus providing the programmer with the required information and procedures needed to structure, design, code, and debug any application program module. The intent of these guidelines was to define a number of practices, procedures and standards that when employed during program production contribute to the overall quality of the program. Specifically the guidelines provide:

- . Design standards which enhance program reliability and maintainability.
- . Recommendations and requirements for card identification, mnemonic names, use of comments and other matters of design and coding which improve program organization and appearance.
- . Procedures for maintaining program correction records.
- . Program debugging plans.
- . Procedures for management of program generation.

The resulting guidelines were presented in detail for the following major sections:

- 1) AN/UYK-7 Computer Operation
- 2) Use of CMS-2
- 3) Use of Common Program
- 4) Program Module Physical Structure
- 5) Program Debugging

CONCLUSION:

The resulting programming guidelines, together with the guidelines for compiling techniques, and CMS-2 user's documentation were the guidelines enforced during the production of the Program 2 application program. These guidelines were followed; in fact, sometimes enforced by the operating system before the coded program module would load and execute.

APPENDIX D

OVERVIEW OF PROGRAM 2 SIMULATION PROGRAM WORKING PAPERS

The Simulation Program Working Papers provided specific supplementary information to the Program 2 Maintenance Investigation Engineering Report to assure the design quality, coding quality, software reliability and maintainability of the Simulation Program for Program 2. These working papers were informal design documents required to commence generation of the simulation program. The overview of each topic covered in these papers is presented in the following paragraphs.

GENERAL SPECIFICATIONS

This paper presented the assumptions, constraints, minimum requirements, and basic operating environments of the Program 2 Simulation Program. The content outline presented was:

- 1) Simulation Program Module Identification
- 2) Program Module Display Capability
- 3) Display Symbol Repertoire
- 4) Tabular Display Format
- 5) Program Module Display Requirements
- 6) I/O Channel and Peripheral Availability
- 7) Minimal Auto/Manual Operating Modes

These brief informal specifications presented a minimal degree of uniformity to the simulation program. This information provided resolution to a number of decisions made important to the design and production of the simulation program, and was not, or intended to be a specification.

SUPPLEMENTARY SOFTWARE STANDARDS AND CONVENTIONS

The standards and conventions covered in this simulation program working paper dealt with:

- 1) Standard Message Type Codes
- 2) Uniform Program Error Processing
- 3) Performance Specification Maintenance
- 4) I/O Coding Standards and Guidelines
- 5) Standard Periodic Structure
- 6) Common Routine Conventions
- 7) Common Data Usage
- 8) Equals Operator Usage
- 9) Deck ID Assignments
- 10) Use of Jointly Declared Variables

These directions supplemented the Program Structure, Compiling Techniques and Programming Guidelines for the operational program as outlined in Appendix B.

MONITOR CONTROL CONSOLE INTERFACE

This working paper supplemented the Monitor Control (MC) Module Program Performance Specification by describing the user program module interface to the MC module. The content of this paper was:

- 1) Display Formats and Uses
- 2) Prestored User Keyboard Entry Data
- 3) Prestored Display Control Specifications
- 4) Message Formats from MC Module to User Module
- 5) Message Formats from User Module to MC Module

MONITOR CONTROL CONSOLE INTERFACE

This information was required for the detailed design and coding of all modules using the MC module. Since the MC module is a part of the developed Program 2 operational program used in the simulation program operating system software, these requirements are a subset of the interface requirements of the MC module of the operational program.

SIMULATION PROGRAM

This working paper provided the necessary direction to simulation program personnel of how the source, object, and listing libraries and system tapes were identified, maintained and transferred between facilities. This information was supplemented in the operational program compiling techniques guidelines for control of the Simulation Program through development, test and maintenance.

SIMULATION PROGRAM SUPPORT SOFTWARE

This working paper supplemented documentation of the simulation program support software by describing:

- 1) Contents of the library tape containing source and object elements of the Program 2 system loader and common program.
- 2) Programmers use information for the Simulation Program System bootstrap loadable tape.
- 3) Adaptation of the Operational System Common Program for use by the Simulation Program.
- 4) Use of the CMS-2 SYSMAKER to generate bootstrap loadable Simulation Program System tapes.

AD-A040 049 SPERRY UNIVAC ST PAUL MINN DEFENSE SYSTEMS DIV
MODERN PROGRAMMING PRACTICES STUDY REPORT. (U)
APR 77 W E BRANNING, D M WILLSON

F/G 9/2

MODERN PROGRAMMING PRACTICES STUDY REPORT.(U)

APR 77 W E BRANNING, D M WILLSON

F30602-76-C-0136

UNCLASSIFIED

PX-11932-F

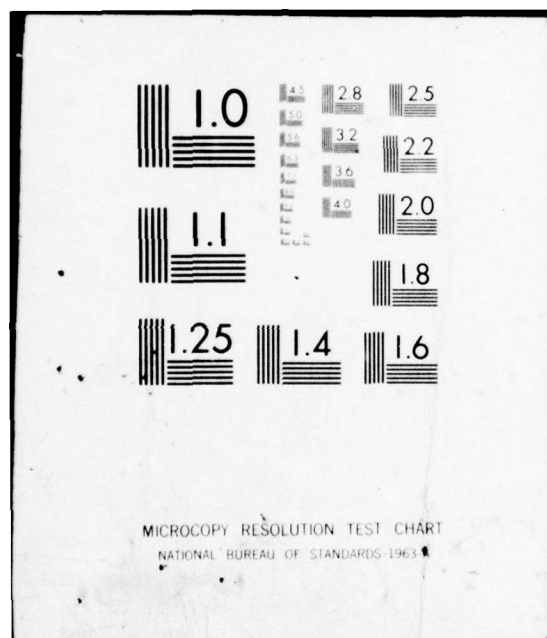
RADC-TR-77-106

NL

5 OF 5
AD
A040049

END

DATE
FILMED
6-77



COMMAND CONSOLE INTERFACE STANDARDIZATION

This working paper briefly describes the guidelines for definition of the naming and use of an array of variable action buttons, digital entry switches, and alternate action buttons on the command consoles.

OPERATOR'S GUIDELINE FOR THE SCENARIO CONTROLLER

The Scenario Controller or scripted scenario generator provided for generation and play back of scenario magnetic tape recordings. These recordings contained timed sequences of events related to simulation of men, machines and environmental data external to but affecting the C&CS Operational Program. This working paper provided a guideline and brief instruction for the use of this simulation tool. Basically, this paper presents the general operating capabilities of the Scenario Controller without providing detailed use in testing and training.

APPENDIX E

OVERVIEW OF THE SYSTEM SOFTWARE MANAGEMENT PLAN FOR PROGRAM 3 AND 4.

OVERVIEW

The development of computer programs for use in the Central Computer Complex was a complex process. Many Navy agencies and their respective contractors were involved in the generation of the computer programs which were integrated for operation in the Central Computer Complex. The large number of independent agencies increased the importance of strengthening and standardizing management visibility and controls in the development. It was also important to establish a sound approach and schedule for communication of information in the planning and design phases, and for the development phases provided for generation, integration test, and certification of the computer programs. This plan emphasized an integration approach conceived to minimize risk inherent in computer program development for the Command and Control System (C&CS). This approach was derived from concurrent schedules established for both computer program generation and integration efforts which provided an orderly transition of development from planning through acceptance and turnover to maintenance, insofar as this is possible within the extremely short time that was available for this work.

Figure E-1 depicts the relationship of this plan to other documents and illustrates how the plan influenced ongoing activities and documentation relating to computer programs.

This C&CS Software Management Plan described:

- a. Responsibilities for performance of tasks defined for C&CS computer program development in relation to the overall management structure for the ship.
- b. The development approach, particularly in the activities associated with computer program generation and integration.

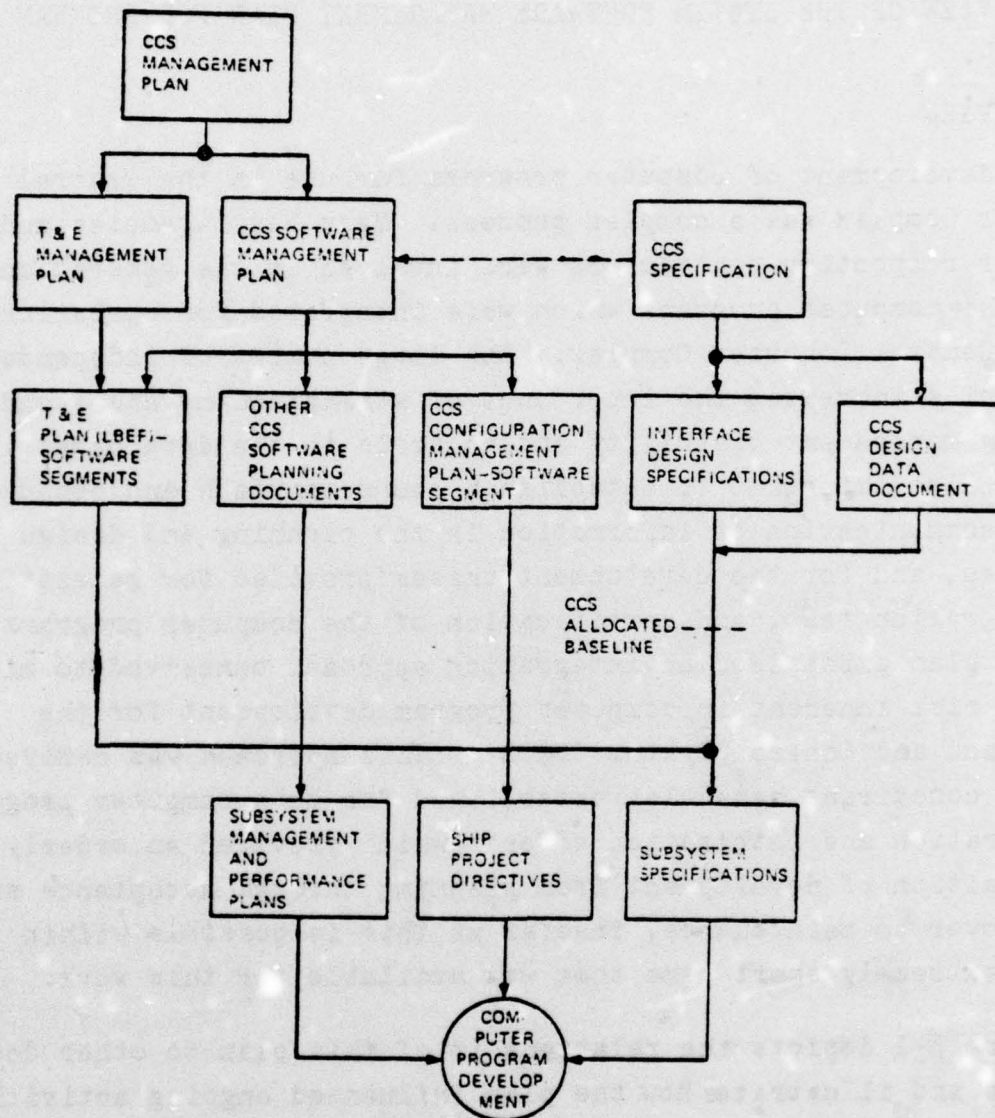


Figure E-1. Relationship of the Software Management Plan with Other Documents

- c. The C&CS in the context of the C&CS and ship subsystem interfaces with the Central Computer Complex, the Central Computer Complex equipment configuration, and the operational software modules of the C&CS.
- d. Computer programs to be required for support of: software development, shipboard operation in the Central Computer Complex, integration test and evaluation of C&CS system hardware and software, and the ongoing mission data analysis and maintenance activities ashore.
- e. The development plan segmented into five overlapping phases including the schedules established for development activities and deliverable items.
- f. Provisioning of computer program development and checkout facilities.
- g. Software development standards and constraints which were imposed on all computer program developers to ensure adequate means of achieving management visibility and controls required for integration, including provision for standard program documentation and development status reports.

APPENDIX F

OVERVIEW OF THE SYSTEM SOFTWARE STANDARDS AND CONVENTIONS FOR PROGRAMS 3 and 4.

OVERVIEW

The development of the Command and Control System (C&CS) software was a complex process. Many Navy agencies and their respective contractors were involved in developing the C&CS programs to be integrated for operation in the Central Computer Complex. Software standardization was required to successfully integrate software and to provide common design constructs for the entire C&CS Operational Program. This document provided C&CS software standardization through formal standards and conventions. Programmers developing the C&CS operational program were required to adhere to these standards and conventions.

This document provided or referenced the base information necessary to generate a C&CS subsystem program which conforms to the required design, coding, and formatting standards and conventions. The detailed sections in this document covered the following topics:

- 1) C&CS System Support Software - including overviews for proper use of the Executive Operating System Software, Compiler Monitor System (CMS-2Y).
- 2) C&CS Subsystem Design Conventions - including descriptions of a subsystem required Regional Control Module and division of a subsystem program into program modules.
- 3) Programming Conventions, Standards, and Techniques including manning conventions for all labels, tape, standards for self-documenting code, and subsystem program code construction standards with examples.

- 4) Coding Conventions - including conventions used when coding in CMS-2 Compiler or Assembler Language, Structured programming concepts, and intended to produce programs which are uniform, readable, and maintainable.
- 5) Coding Techniques - including techniques, hints and methods for efficient code, coding limitations, approaches for use of libraries and methods of increasing job throughput.
- 6) Flowchart Conventions - describing use of the American National Standard figures when program flow charts are provided.
- 7) Glossary - including terms generally used by the C&CS subsystem programmer.

The initial edition of this document was followed during the development of Programs 3 and 4, but the revised edition was distributed too late to effect the design and coding of Programs 3 and 4. The application of topics 3, 4, and 5 was a tedious process for the programmer, but the uniformly coded result was well worth the effort. The result was an accurately self-documented program listing that was consistent and easily read by agencies other than the developer.

APPENDIX G

OVERVIEW OF THE EXECUTIVE OPERATING SYSTEM USER'S REFERENCE MANUAL FOR PROGRAMS 3 AND 4.

The Executive Operating System User's Reference Manual evolved from the Common Program Guidelines document from Program 1. The Executive Operating System User's Reference Manual was prepared by the Command and Control System (C&CS) integration contractor with technical support of the group developing Program 3. The reason for this support was two-fold: (1) the Executive Operating System program was formed by integrating the Navy refurnished Common Program with Program 3, and (2) the Navy refurnished Common Program was derived from the Common Program of Program 1.

This User's Reference Manual defined or referenced definitions of all software interfaces, data, and logic required by C&CS subsystem programmers necessary to design and code program modules for operation with the Executive Operating System. The detailed sections in this document covered the following topics:

- 1) Executive Operating System Program Overview.
- 2) Relation of other C&CS documents to this manual.
- 3) Requirements imposed on C&CS subsystem including:
 - a) Dividing Subsystem Programs into Modules
 - b) Module Structure
 - c) Module Execution Requirements
- 4) The interfaces and ground rules for using services of the Executive Operating System including:
 - a) Executive Service Requests - program callable execute control and subroutines.
 - b) Program Module Entrances - executive call for task processing featuring six distinct scheduling characteristics.
 - c) Time-slicing - executive feature for three program module entrances.
 - d) Program Module Control Interfaces - executive provides for activation, deactivation and reconfiguration.
 - e) Program Module Interfaces - executive supports shared data areas and intermodule messages.
 - f) Intercomputer Data Transfer - executive provides the C&CS Computer A/Computer B interface.

- g) Input/Output - use of executive interfaced standard peripherals and registration for use of non-standard interfaced devices.
 - h) Program Segment Manipulation - Load, relocate, delete.
 - i) Common System Routines - Math, conversion, data handling and confidence tests.
 - j) System Time - software real-time clock and time of day.
5. General guidelines for design of a subsystem program including:
- a) Features of region controller modules - program load, computer residency, memory management, region data base, region routines, and error handling.
 - b) Input/Output Programming Guidelines.
 - c) Real-time On-line Debugging.

This manual was prepared and delivered as preliminary information to all C&CS subsystem program developers by the system integration agency. The preliminary manual was in error on many prescribed interfaces, but did provide a basic outline of functions of the developing Executive Operating System Software. Through review by the group developing Program 3 and the subsystem program developers, including the group developing Program 4, this manual was revised and reissued to correctly describe the final released Executive Operating System software for use by the subsystem program developers.

By the time the revised manual was released, Program 4 was completed. Therefore, the manual was no advantage for development of Program 4. However, the revised manual has been valuable for development of all other subsystem programs and for the continued maintenance of the C&CS software.

APPENDIX H
GLOSSARY OF TERMS

ABSOLUTIZING

Creating loadable object machine language code.

ANNOTATED LISTING

Printer or written comments on a computer generated printer output listing.

APPLICATION PROGRAM

Program specifically produced for a particular use of the computer system. Application programs usually require an operating system for general control. Usually an application program consists of a net of integrated tasks to provide a primary system function such as Navigation or Gun Fire Control.

AUTOMATED TOOLS

Used in reference to the software capabilities available on the UNIVAC 1108 computer system (e.g.), text editor, etc.).

AUTOMATIC CASUALTY RECOVERY

Casualty recovery within a system that requires no manual input to complete.

GLOSSARY OF TERMS (continued)

BACKGROUND (MODE)

A condition of executing a computer program in a multiprogrammed system usually under control of an operating system such that the computer resources used for the execution are not required for any other higher priority operation.

BASELINE PROGRAM

A program possessing well defined capabilities and functions which is decreed to be the starting point for further program development.

BOTTOM-UP PROGRAM TESTING

A method of testing where the independent functions are tested first and are then used to test the higher level dependent functions.

BOTTOM-UP PROGRAMMING

Generation of a program as an isolated building block. Necessary independent subprograms are generated first followed by the dependent functions.

BUILDING BLOCK (See Bottom-Up Programming)

GLOSSARY OF TERMS (continued)

CASUALTY

Failure of operational status of a hardware device or component.

CASUALTY RECOVERY

The capability for a system to resume full or degraded operation after experiencing a malfunction during operation.

CERTIFICATION PLAN

An approved document containing a plan to provide software certifications.

CERTIFICATION TEST

The formal demonstration to the customer of the tests documented in the certification test procedure.

CERTIFICATION TEST PROCEDURES

A formal document detailing the actions to be performed and results to be observed in verifying the correct operation of a program.

CHECKOUT LOG

A record kept during program testing of all changes made to a program.

CHECKOUT PLAN

A plan of action for testing a program. It is written down on either an informal or formal form.

CHECKOUT PROCEDURE

A detailed breakdown of each step in a checkout plan. It details each action to be performed in testing a particular function.

CHECKOUT TEST

An informally defined process taken by the programmer to ensure the desired and specified proper operation of his developed program involving debugging and execution of the code.

GLOSSARY OF TERMS (continued)

CHECKOUT TESTING

The testing performed on software usually by the programmer of each programmed element of a subprogram (design object of a program module) to ensure the program operates as specified and as intended by design.

CLOSED-LOOP TESTING

The testing process where the testing routine accepts input data, processes the data to generate results which are both made available to the operator for his analysis, and used to calculate corrective feedback data to affect the next input data.

CMS-2Y

The Navy's standard compiler/monitor/librarian system for use on the AN/UYK-7 computer for code generation for the AN/UYK-7 computer.

COMMON PROGRAM (CP)

Navy standardized realtime operating system software for applying the AN/UYK-7 computer to various military applications. The Common Program includes the program modules called: Standard Executive, Common Peripheral, Common System, Debug Module, Dynamic Replacement, Confidence Test, and Utility Package.

COMMUNICATION LINK

A channel to transfer information.

COMPUTER FUNCTIONAL HARDWARE COMPONENT

Hardware element of a computer that is removeable and interchangeable without disassembly and also directly affected in operation by software. For the AN/UYK-7 computer these are CPUs, IOCs, IOAs and MMUs.

COMPUTER RESOURCES

The total of computer capabilities, memory, and mass storage.

GLOSSARY OF TERMS (continued)

COMPUTER PERIPHERALS

The set of equipment associated with a computer which provides input, output, mass storage, and communication capabilities.

CONFIDENCE TEST

A program which verifies the proper operation of a hardware device or system function.

CONFIGURATION

The total software modules in a software system or hardware devices in a hardware system, and their interrelationships.

CONFIGURATION CONTROL

A process by which a configuration item is baselined, and thereafter, only changeable by approval by a controlling agency. For example, the final program performance specifications on a program may be placed under Navy configuration control after initial Navy approval.

CONFIGURATION MANAGEMENT

A discipline applying technical and administrative direction and surveillance to identify and document a configuration item, to control changes to it, and to report status of change processing and implementation.

CONTROL MEMORY

Digital memory used in computer processors and input/output controllers for their internal control, computations, logic and status represented as binary data.

GLOSSARY OF TERMS (continued)

CONVENTION

An agreed method, form of presentation to provide consistency and understanding to deliverable software elements.

CONVENTIONAL PROGRAM TESTING

This testing method prescribes the approaches and considerations for the testing of a program as a "completed" isolated building block.

CORE

A form of digital memory using magnetically bistable donut shaped devices which are electronically set, changed and sensed. Sometimes used in place of the term "main computer memory."

CORE RESIDENT

Referring to software code which ordinarily resides in the computer's memory during system operation.

COST EFFECTIVE

A term which describes some method, tool or technique that reduces the predicted cost to perform on a contracted item.

GLOSSARY OF TERMS (continued)

DATA EXTRACTION

A software test support tool with the capability to extract specified data at specified times in on-line operation. The data is written to a magnetic tape.

DATA REDUCTION

The capability to print the data from the data extraction tape in a format specified by the operator.

DEBUG

To verify that the executable program code functions properly.

DEBUG AIDS

A set of software tools available to the system operator and used to locate errors in software. (It includes dump, snap, inspect and change, and time capabilities.)

DEPENDENT FUNCTION

In software, a program which depends on other programs or subprograms to implement some of its functions.

DESIGN HIERARCHY

In software, a program design in which the identified programmable elements are arranged in order of dependency from the most dependent elements to the least dependent elements.

DESIGN OBJECT

A construct used in software design, production and test to identify the location of each coded object in accordance to a list of rules.

DESIGN OBJECT NETWORK

A drawing depicting the hierarchy and real interconnections of design objects with a subprogram.

GLOSSARY OF TERMS (continued)

DESIGN OBJECTIVE

Goals to be accomplished by the software being generated.

DESIGN REVIEW

A scheduled meeting between customer and manufacturer to determine that a proposed software configuration will satisfy approved performance specifications.

DESIGN SPECIFICATION

A document containing the approved design requirements for a program.

DEVELOPMENT MODEL

Program ready for use during development of the operational program.

DEVELOPMENT SCHEDULE

The time frame with its associated milestones for a program or system being developed.

DIAGNOSTIC TESTS

Test programs for operation with the hardware device being tested which will identify any defect of operation from the specification and will identify the probable failure to some group of replaceable elements.

DYNAMIC TESTING

To test a program or subprogram as part of a software system while subjecting it with a real or simulated environment.

GLOSSARY OF TERMS (continued)

ELEMENT CHARACTERISTICS

Those actions, functions, and interfaces peculiar to a program element.

ERRATA

Changes in software intended to correct errors.

EXECUTIVE (PROGRAM)

A centralized subprogram which provides system use functions such as control of input/output and scheduling of use of the processor(s) and allocation of computer resources including processing time, memory access, and interrupt processor services required by application subprograms and their programmed tasks.

EXECUTIVE STATE

A mode of operation within the computer processor which allows for software to control all computing operations using a dual state (executive over task) architecture.

GLOSSARY OF TERMS (continued)

FAULT TOLERANCE

The capability to withstand malfunction faults or failure during operation. Specifically in this study, fault tolerance consists for fault detection, isolation of a fault to a single component, and recovery to resume operation without use of the failed component.

FIRMWARE

An executable digital program which is fixed in the memory of the computing device which will execute it. For example, a program in the AN/UYK-7 CPU's NDRO is firmware.

FLOWCHARTS

An automated flow chart generator program.

FULL SIMULATION

Using simulation software to replace any missing elements in the operational environment.

FULL LOAD ENDURANCE TEST

A test performed continuously over a period of several days processing input data and stimulating the tested elements at a maximum rate.

FUNCTION

The natural, required, or expected activity of a program element in carrying out a program requirement.

FUNCTIONAL CERTIFICATION

(See software certification).

GLOSSARY OF TERMS (continued)

GROUND RULE

A basic rule which must be followed in order to attain the proper use or design of a software element.

GUIDELINE

A recommended method or example procedure to be followed for proper use and design of a software element.

HARDWARE

Physical equipment, e.g. mechanical, electrical, or electronic devices (contrasted with software).

HARDWARE SUITE

A set of integrated hardware devices available for use at a specific location.

HIGH LEVEL TESTING

To test a software system as a whole.

HOST SOFTWARE

Software normally provided for one computer which is actually being run on another computer. For example, program modules for use with the Common Program and AN/UYK-7 computer can be run using host software of the AN/UYK-7 simulator program on the UNIVAC 1108 system.

GLOSSARY OF TERMS (continued)

INDEPENDENT FUNCTION

In software, a subprogram module which makes no calls to any other lower level module, therefore, is independent of changes made in other software.

INDEPENDENT TESTING

The capability to isolate the function under test from other system processing.

INFORMAL DOCUMENTATION

Written matter produced during the course of work on a contract, but non-deliverable and often non-standard such as engineering notes and records.

INFORMAL TESTING

To test using informal documentation. Usually a preliminary form of testing performed before a formal certification test.

INTER COMPUTER

Communication between two or more computers usually using a special input and output channel where the peripheral is another computer.

INTERACTION

The reciprocal effects of one software module or hardware device on another.

INTERFACE

The common boundary between software modules, between hardware devices, or between software and hardware.

INTERFACE TEST

An executable routine, sometimes a computer program, which tests the communication paths between integrated functional hardware components.

GLOSSARY OF TERMS (continued)

INTERFACE TESTS

Test routines which stimulate use of some digital interface between integrated hardware devices specifically to detect a failure for response. For example, software executing in a CPU can request reading and writing of data and execution of a computer word from each of the MMUs integrated by an instruction or operand buss in the AN/UYK-7 to test the CPU/MMU interface.

INTERMODULE MESSAGE

A group of computer words containing information to be transferred from one program module to another. This capability is provided by the message handler of the common program executive.

INTERRUPT

A computer hardware generated signal which causes suspension of a current CPU execution sequence, and initiates CPU execution of another sequence.

INTERRUPT STATUS CODE

In the AN/UYK-7 computer's CPU, the interrupt status code is a 10-bit code identifying the interrupt by type and source.

INTRA COMPUTER

Communications between modules executing in the source computer.

I/O INTERFACE SOFTWARE

Programs which generalize the input/output of data and control signals between peripheral devices and application programs.

GLOSSARY OF TERMS (continued)

LAND BASED OPERATIONAL TESTING

Software testing performed at the LBEF.

LEVEL OF TESTING

Degree of thoroughness desired in determining the correctness of software and how well it carries out its' specified requirements.

LEVELLED DEVELOPMENT

The process of baselining code before it is fully developed, to allow for early use and software integration.

LOCAL DATA

Programmed data element required by the task operation of a single program module or subprogram.

LOCAL DATA PROCESSING

Engineering data processing controlled by an operator at the computers' site.

LOW LEVEL TESTING

To test individual subprograms or programs elements within a subprogram.

GLOSSARY OF TERMS (continued)

MANPOWER SCHEDULE

The allocated or resulting engineer manning on a contracted project present for the contracted period on a monthly or weekly basis.

MAPPER

A software tool which provides for management of information and report generation.

MILESTONES

An established event set to occur at a specific time in the schedule of completing a larger task used for production planning and identical of project status on contracted work items.

MOD X2 COMMON PROGRAM

The Navy's baseline Common Program for realtime applications of the AN/UYK-7 Computer.

MOD X2

Model X (experimental) 2 (revision)

MOD X2 STANDARD EXECUTIVE

The executive program module of the MOD X2 Common Program under formal Navy configuration control.

MODELLING

To simulate or represent an item or technique via software.

MODULAR CONSTRUCTION

To construct a software program or system of modules.

MODULE (SOFTWARE OR PROGRAM)

A self-contained subprogram that satisfied some processing requirements of a particular set of function and consisted of data stores and program logic which is reorganized by the operating system executive having executable tasks.

GLOSSARY OF TERMS (continued)

MULTIPLE PROCESSOR

A computer which contains two or more central processing units (CPUs).

MULTIPROCESSING

A program execution that allows for simultaneous execution of a shared copy of a coded element by two or more CPUs.

MULTIPROCESSING SOFTWARE

Software designed to allow simultaneous sharing of the program execution by more than one processor.

MULTIPROGRAMMED

Software design allowing simultaneous execution of many tasks satisfying the functional needs of the system as though each acts independently.

MULTIPROGRAMMING

A program design that allows support of many functions simultaneously as though each function operates alone. Multiprogramming is provided for the AN/UYK-7 computer by proper use of the Common Program Executive.

GLOSSARY OF TERMS (continued)

ON-LINE DEBUGGING

The process by which a programmer uses software tools to operate a program in a very controlled manner to perform checkout testing, identification of any erroneous operation and correction of software errors, and these tools are available at terminals on-line to the operating system.

ON-LINE TESTING

See On-line Debugging.

ONE-SHOT TESTING

Usually associated with bottom-up testing where the testing is performed only on the completed program.

OPEN-LOOP TESTING

The testing process where the testing routine accepts input data, processes the data to generate results which are then made available to the operator for his analysis.

OPERABILITY TEST

A test run before starting or during initialization of an integrated system such as a digital computer complex, which identifies any abnormal operational status of any device or interface between devices to determine the operational status of the equipment.

OPERABILITY TESTING

A test run to sample functions of an operational system such that every element of the hardware/software system is shown to be intact and suitable for operation.

OPERATIONAL PROGRAM

The program which provides for execution during the real application of the digital system. For example, the integrated system of the operating system software and the application subsystem programs may form the operational program for a realtime control system.

GLOSSARY OF TERMS (continued)

OPERATING SYSTEM

A digital hardware/software system which provides the environment for operating task application programs.

OPERATING SYSTEM SOFTWARE

Software of an operating system that includes the executive, peripheral interfaces, common subroutines, debug aids, system utilities and other general software supporting task application program execution. Usually the operating system software provides for multiprogramming and multiprocessing of application program tasks.

OPERATIONAL ENVIRONMENT

The set of all external stimulus and data sources with which a software system interfaces and communicates.

OPERATIONAL EVALUATION

The analysis of a system operating in its' real life environment.

OPERATIONAL TESTING

Performing tests on software in its normal operating environment.

GLOSSARY OF TERMS (continued)

PATCH

An object coded program change written in machine code, and inserted to overlay language produced object code for temporary repair of a coding error or program discrepancy until the coded element is reprocessed and regenerated as a source code change.

PERFORMANCE SPECIFICATION

The official document containing the performance requirements for a program.

PERFORMANCE REQUIREMENTS

The specified set of goals which must be attained by a software system.

POFA (Performance Operability Functional Analysis)

A computer program capable of determining the operational performance of functions for a specific device.

PREAMBLE

An element of a program module which identifies the tasks and task parameters of the program module for use by the executive program.

PROCESSOR OR CENTRAL PROCESSING UNIT (CPU)

Hardware unit containing execution control, logic and arithmetic computation, instruction interpretation and data manipulation.

PROGRAM

Effort expended under contract that results in a documented, coded and tested software element for use in a digital data computer.

PROGRAM ACCEPTANCE

Official demonstration to the customer that the software performs according to approved specification and acknowledgement of this fact by the customer.

GLOSSARY OF TERMS (continued)

PROGRAM ANALYSIS

The period during which a study is made of the feasibility and requirements of a proposed software solution to a problem.

PROGRAM CHECKOUT

(See Program Testing).

PROGRAM DELIVERY

Delivery of a tape(s) containing source and object code of the program to the customers site.

PROGRAM DESIGN

The period during which detailed plans and procedures are determined to implement the software proposed in the program analysis period.

PROGRAM DEVELOPMENT

The generation of software from its original conception through analysis, code production, checkout, documentation, delivery, and integration of the completed program.

PROGRAM ELEMENT

Any software unit.

PROGRAM ENVIRONMENT

(See Operational Environment)

PROGRAM EVALUATION

The process of studying a program to determine how well it fulfills its designed purpose.

PROGRAM INTEGRATION

The combining of subprograms into subsystems or subsystems into the whole programmed system, thus resulting in the whole program specified for operation within the digital computer equipments.

PROGRAM LEVEL

A subset of modules selected from a top-down hierarchical design. (e.g. it consists of all functions from the top down to a specified "level".)

GLOSSARY OF TERMS (continued)

PROGRAM LIBRARY

A controlled set of all documentation, baseline programs, new programmed elements, support software and tools available for development and configuration management of a program.

PROGRAM MAINTENANCE

The process of making corrections, additions, or modifications to software.

PROGRAM MODIFICATION

To change correct, or add capabilities to software.

PROGRAM MODULE (See module)

PROGRAM PRODUCTION

The period during which code and documentation are generated for the program.

PROGRAM SEGMENT OR SEGMENT

The smallest coded unit of a program which can be loaded as one logical entity. For example, the entity loaded for access by a task base register of the AN/UYK-7 is called a program segment.

PROGRAM STRUCTURE

The arrangement or interrelation of all the parts of a program or software system.

PROGRAM TESTING

The period during which the software is checked out to determine that it fulfills all requirements specified for it.

PROGRAM TESTING PERSONNEL

Usually members of an autonomous test group.

GLOSSARY OF TERMS (continued)

PROGRAM UPDATE

(See Program Maintenance).

PROGRAMMING CONVENTIONS

(See Convention)

PROGRAMMING STANDARDS

(See Software Development Standards)

PROJECT

An undertaking whose goal is the fulfillment of a contract or related contract.

PROJECT ENGINEER

A person assigned to the overall technical responsibilities on a contract in Sperry Univac.

PROJECT PLAN

Internally controlled formal document required on all projects which defines to company management the contract form, the statement of work, the milestones schedule, the method of completing all delivered items, evaluation of risk, and customer relations.

GLOSSARY OF TERMS (continued)

QUALIFICATION TESTING

Qualification test of software consists of performing a controlled execution of a deliverable program package such that all specified realtime and functional requirements are known to be satisfied by the program package. Normally, this involves: documentation describing the test (test plan and test procedures) approved by the customer through a review process; generation of a test system (hardware and software) to provide the simulated environment, test controller and data recorder; and customer observed operation of the test and compilation of the test results.

QUALITY ASSURANCE

A process by which all deliverable items are formally checked to assurance their quality in accordance to customer approved standards, methods and specifications.

GLOSSARY OF TERMS (continued)

REALTIME

See Realtime Programming

REALTIME AUTOMATIC CASUALTY RECOVERY

A process by which realtime operation of a system recovers automatically upon occurrence of a hardware failure within the system.

REALTIME AUTOMATIC CASUALTY RECOVERY

The capability for a system to automatically resume full or degraded operation automatically after experiencing a malfunction during operation, and to do so in such a manner and within a specified time period such that the system support critical to a mission is not lost.

REALTIME ON-LINE MAINTENANCE TESTING

A method of operating hardware test software on-line with the operational programs without degenerating realtime properties of the system. For example, the AN/UYK-7 diagnostic program can be operated using a MMU, CPU and IOC while the operational program uses the remaining MMUs, CPU & IOCs of a multiprocessor AN/UYK-7 computer.

REALTIME PROGRAMMING

Program development (analysis, design, production, and testing) resulting in programs which monitor and control events within the time constraints of the computers external environment. An example of these time constraints are those imposed by sensors, displays, weapons and other online equipment.

RECABLING

Exchanging physical cables between one hardware device and another to obtain a different system configuration.

GLOSSARY OF TERMS (continued)

RECONFIGURATION

The process of reassigning hardware devices available to the operational program and/or redefining the available functions of the operational program. For example, on Program 1, if a memory unit is identified as unusable by the operational program, all programs formerly using the unusable unit are allocated and loaded in some other area of memory reserved for such an occurrence and normal operation continued with a reinitialization of the program modules reconfigured to a new memory area.

REGIONAL DATA

Programmed data element required by the task operation of two or more program modules (subprograms) within one application program (subsystem).

REMOTE DATA PROCESSING

Engineering data processing controlled by an operator at a terminal not located at the computers site, but channeled to the computers site.

REPEATABILITY

A property required of software tests, that each time they are executed, the results will be the same.

RESUME

A condition in which the hardware logic detects a request made along a digital transfer buss in a computer has exceeded a specified gross time period.

GLOSSARY OF TERMS (continued)

SCENARIO

An automated test control package consisting of test execution control words, test data, and event stimuli used to activate and test a target program.

SCHEDULING

A process of the executive program by which use of the CPU is allocated to task programs.

SCOPE

The range or extent of application (e.g. the extent to which software is tested.)

SCRIPTED SCENARIO

A capability of entering a scenario from some digital input device, such as a display console, to control simulation and simultaneously recording, such as on magnetic tape, the scenario such that the scenario may be saved and later automatically played back.

SEA TEST MODEL

Program baseline for use during sea trials.

SEGMENT LINKAGE TABLE

A data table of a program module which identifies all segments to be accessible to the tasks of the module.

SEGMENTED PROGRAM

A program subdivided into several blocks of code, where each block may be loaded independently and referenced by a different AN/UYK-7 base register.

SIMULATION

The practice of emulating the presence and actions of the interface normally presented by a software module or hardware device.

SIMULATION MODULE

A program whose purpose is to imitate the interface of other software modules or hardware device.

GLOSSARY OF TERMS (continued)

SIMULATION PROGRAM

A program which provides the appearance of a live realtime environment for use in testing an operational program or for training the crew's use of an operational program.

SOFTWARE

The collection of data coded as program logic, routines, data and information for use with a computer (e.g., application or operating system software as contrasted with hardware).

SOFTWARE CERTIFICATION

The process which formally demonstrates to an independent agency of the customer that the delivered program does indeed meet all specified performance requests.

SOFTWARE DEFICIENCY

A missing function or capability required in software.

SOFTWARE DESIGN

A description of how software will be produced to satisfy the software specification.

SOFTWARE DEVELOPMENT

The process of analysis, design, production, test and continued support to provide software.

SOFTWARE DEVELOPMENT INSTRUCTIONS

Informally documented internally applied instructions to programmers to improve and simplify program design, generation and maintenance.

SOFTWARE DEVELOPMENT STANDARDS

Standards, conventions, ground rules, guidelines, procedures, and software tools employed during the software development process to benefit software design quality, coding quality, software reliability, viability and maintainability.

GLOSSARY OF TERMS (continued)

SOFTWARE ENGINEER

The term used in this study to identify either a programmer or system analyst.

SOFTWARE INTEGRATION

(See Program Integration).

SOFTWARE MODULES

Self-contained programs that satisfy some processing requirements of a particular function consisting of data stores and program logic and special executive interface data and logic.

SOFTWARE SPECIFICATION

Specification of what software to be developed will do functionally.

SOFTWARE SYSTEM

(See System).

SOFTWARE TOOL

A programmed element which is useful in follow-on software development.

SOFTWARE VERIFICATION

The process by which software programs have been executed in a testing condition to assure a degree of satisfaction of specified requirements.

STANDARD

A level or grade of excellence attainment, etc., regarded as a goal or measure of adequacy.

STATIC TESTING

To test a program or subprogram as an individual entity and not as part of a system.

GLOSSARY OF TERMS (continued)

STRUCTURED NARRATIVE

The procedure of writing a design specification as a set of steps explaining the operation of the program. Each step uses a formal construct with data pertinent to the program.

STUBS

A "dummy" software element used in place of an expected functional element until the expected element becomes available. A stub can satisfy interface requirements, trace execution flow, and model program behavior by consuming CPU time, occupying memory space, and using mass memory.

SUBPROGRAM

A collection of program elements which together provide a function or relatively independent functions with respect to the whole program (also see program module).

SUBSYSTEM

A collection of subprograms which together provides a major function and is independent of any other subsystem. The navigation program on Program 1 is an example.

SUPPORT

To furnish skilled programmer/analysts to aid the customer in Program Integration and/or Program Maintenance.

SUPPORT SOFTWARE

Software tools used by project personnel for software design, debugging, testing, verification, and management.

SUPPORT TOOLS

The set of software tools and procedures used as aids in software development.

GLOSSARY OF TERMS (continued)

SYSDMAKER

The system tape generation and utility program available in CMS-2Y for maintenance of bootstrapable system program tapes for the AN/UYK-7 computer. For example, the system tape for CMS-2Y is maintained by SYSDMAKER and is sometimes called a SYSDMAKER (generated) tape.

SYSTEM

A set or arrangement of software or hardware so related or connected as to form a unity capable of achieving the goals specified in its design.

SYSTEM DATA

Programmed data element accessible to all tasks of any program module, i.e. global data. For example, data shared by two or more application programs (subsystems) is called system data.

SYSTEM DESIGN

The process of translating the base system requirements into a set of detailed performance specifications including finalization, with customer concurrence of the system hardware and software structure.

SYSTEM GENERATION

The process of combining elements of machine executable code into a stand alone executable program usually by making the proper linkage of programmed tasks and segments of code with the operating system and recording on a mass memory device (magnetic tape or disk).

SYSTEM INTEGRATION

The process of combining physically, electronically, and functionally all elements specified for a system, usually composed of both hardware and software. For example, system integration is performed at an evaluation facility before acceptance testing may begin.

SYSTEM VIABILITY

The capability of software to evolve into or remain a useful commodity on many projects. (e.g. Common Program Operating System).

GLOSSARY OF TERMS (continued)

TARGET DEVICE

The device matching the hardware device for use in the final end product operational hardware configuration.

TARGET MACHINE

(See Target Device).

Task

A program sub division which provides an executable form to accomplish a specific function.

TASK STATE

One or two states or modes of operation by the CPU of the AN/UYK-7 computer. The task state is restricted from using privileged instructions and is the normal state for non executive programs such as an application program.

TECHNIQUE

The method used in carrying out some operation.

TEST CONTROL PROGRAM

Usually refers to scenario or the program reading and interpreting a scenario, but could be any software which provides automatic direction for testing.

TEST MODULE DRIVER

Independent modules which execute under the same operating system as the software to be tested and perform specific tests in response to external stimuli.

TEST PLAN

A formal document which defines the tests to be performed to verify that the computer program meets the performance requirements stated as the acceptance criteria defined in the Program Performance Specification.

GLOSSARY OF TERMS (continued)

TEST PROCEDURE

A formal document developed from a Test Plan that presents detailed instructions for the set up, operation, and evaluation results for each defined test.

TEST PROGRAM

A program which verifies the proper operation of an element, or system of elements of hardware or software, both hardware and software.

TEST REPORT

A document recording the results of a program test.

TEST SCENARIO

(See Scenario)

TEST SOFTWARE

(See Test Program)

TEST SYSTEM

The system of hardware, operational software and test software integrated and used to run product acceptance tests on the operational software.

TEST SYSTEM VIABILITY

The capability of test software to remain or evolve into a useful tool of value to many projects or programs.

TEST TOOLS

The support software used as an aid in program checkout and debugging.

TEST VALIDITY

The degree to which a test accomplishes its specified goal.

TESTING CRITERIA

The requirements the program under test must satisfy.

GLOSSARY OF TERMS (continued)

TESTING EFFECTIVENESS

The degree to which the software can be checked out with all "bugs" removed.

TESTING METHODS

The procedures and tools utilized in proving that a program correctly fulfills its requirements.

TESTING OBJECTIVE

A goal to be attained from testing software.

TIME COMPRESSION

The exclusion from a sequential set of time intervals all those for which some desired action did not occur. (e.g. A tape is created with one record generated each second. The records are blank unless a specified key press occurred. In time compression the blank records would be eliminated from the tape. Playing this tape back results in a speeded up reproduction of the specified actions).

TIME SERIAL

A progression of actions, data input or data output in consecutive equal time intervals, such as a magnetic tape recording of operator actions in each second.

TOP DOWN

A general term indicating a hierarchy of dependent elements and an order of analysis, definition, design or production beginning with the most common element(s) (Top) to the least common elements (on Down).

TOP DOWN PROGRAM TESTING

This testing method prescribes the approaches and consideration for the testing of a program as an "evolving" isolated building block

TRANSPORTABILITY

The capability of a program to operate on various different make and model computers with a minimum of modification required.

GLOSSARY OF TERMS (continued)

ULTRA/32

The AN/UYK-7 computer assembly language or language processor. Also, the assembler language available in CMS-2Y.

ULTRA COMPUTER

Communication with devices external to the computer(s).

VERIFICATION

To assure that a program carries out its specified requirements. (See Software Verification).

WORK BREAKDOWN STRUCTURE

A management tool used on projects and proposals to identify the individual cost centers for assigning costs and maintaining control on all individual items of work or a project.

WRAP AROUND SIMULATION

Simulation of the entire real environment including all external equipment and other software systems supplying data to the computer system. Standard peripherals such as disks, printers, etc. are considered part of the internal computer system in this case.

APPENDIX I
BIBLIOGRAPHY

SSN-688

1. NAVSHIPS 0967-029-6710, Program Performance Specification for the SSN-688 Navigation System Operational Computer Program (confidential) Sperry Univac DSD, 20 November 1973, Naval Ships Systems Command.
2. NAVSHIPS 0967-029-5260, Software Guidelines for Users of FY69/70 SSN Central Computer Complex Common Program, 3 March 1970, Naval Ships Systems Command.
3. NAVSEA 0967-LP-029-5250 Revision A SSN 688 Central Computer Complex Operating System Program Specification (Confidential), Sperry Univac DSD, 1 March 1976, Naval Sea Systems Commands.
4. NAVSHIPS 0967-027-6080, Computer Program Certification Procedures for SSN-688 Navigation System Operational Computer Program, Autonetics Div. of GD, 20 December 1974, Naval Ships Systems.
5. Contract No. N00024-72-C-5064, Computer Program Test Report for SSN-688 Navigation System Operational Computer Program, Autonetics Div. of GD, 11 April 1973, Naval Ships Systems Command.
6. UNIVAC PX 10444, SSN 688 Class Ship Systems Manual - Central Computer Complex Description and Operation, Volume 2, Chapter 6, 28 Sept. 1976, Sperry Univac Naval Systems Div.
7. NAVSHIPS 0967-027-5320 Part 2, Program Certification Procedures for SSN-688 Common Program, Sperry Univac DSD, 15 May 1973, Naval Ships Systems Command.
8. NAVSHIPS 0967-027-5320 Part 1, Program Certification Plan for SSN-688 Common Program, Sperry Univac DSD, 18 Dec 1970, Naval Ships Systems Command.

SSN-688 (continued)

9. PMS 393-CCC-TN1, SSN-688 Class Central Computer Complex Interim Software Maintenance Configuration Management Plan, Navy, 1 Oct. 1975, Navy
10. NAVSHIPS 0967-029-5710, Configuration Management Plan for SSN-688 Central Computer Complex Software, Navy, 1 Oct. 1971, Naval Ships System Command.
11. PMS 393-CCC-TN2, SSN-688 Class Central Computer Complex Interim Software Maintenance Problem Reporting instructions, Navy, 1 Oct. 1975, Navy.
12. UNIVAC PX 6171, SSN-688 Central Computer Complex Software Development and Integration Configuration Management Adjunct Report, Sperry Univac, 6 Nov. 1970, Sperry Univac DSD.
13. Sperry Univac PX 7805, Program Description for SSN-688 Navigation Program Navigation Training/Simulation Module, Sperry Univac DSD, 15 June 1972, Sperry Univac DSD.
14. Sperry Univac 7874, Operating Procedure for SSN-688 Navigation Program Simulation Program, Sperry Univac DSD, 14 Aug 1972, Sperry Univac DSD.
15. NAVSEA 0967-LP-027-7890, Configuration Management Plan for SSN-700/594 Central Computer Complex Software, Sperry Univac DSD, 1 Sept. 1976, Naval Sea Systems Command.

DLGN-38

1. Contract No. N00024-69-C-1233, DLGN-38 Program Maintenance Investigation Engineering Report, Naval Systems Document:
 - Part 1 - Summary, 26 June 1972.
 - Part 2 - DLGN-38 NAVMAT Command & Control Operational Program Documentation Guidelines, Revision 1, 18 June 1973.
 - Part 3 - DLGN-38 Input/Output Formatters for the Command & Control Operational Program, 26 June 1972.
 - Part 4 - DLGN-38 Program Structure for the Command & Control Operational Program, 26 June 1972.
 - Part 5 - DLGN-38 Compiling Techniques for the Command & Control Operational Program, 26 June 1972.
 - Part 6 - DLGN-38 Programming Practices for the Command & Control Operational Program, 26 June 1972,
2. NAVSEA, Integrated Combat System Management Plan for DLGN-38 Class, Revision 5, January 1975, Naval Sea Systems Command.
3. NAVSEA, DLGN-38 Command and Control System Computer Program Development Plan (confidential), October 1972, Naval Sea Systems Command.
4. NAVSEA 0967-LP-029-6220, Computer Program Design Specification for DLGN-38 Class C&CS Operational Program, paragraph 3.3.2.3- System Loader Module, 17 May 1976, Naval Sea Systems Command.
5. NAVSHIPS 0967-029-6210, Computer Program Performance Specification for DLGN-38 Class C&CS Operational Program, paragraph 3.4.3 - Dynamic System Reconfiguration, 1 May 1975, Naval Ships Systems Command.

DLGN-38 (continued)

6. NAVSEA 0967-LP-014-1081, DLGN-38 Ship Operational Readiness Test Plan for AN/UYK-7 Computer Complex C&CS and SIDS, 22 July 1975, Naval Sea Systems Command.
7. NAVSEA 0967-LP-011-2240, Operating Procedure for DLGN-38 Class Utility Package for use with AN/UYK-7 Computer, 16 March 1975, Naval Sea Systems Command.
8. DLGN-38 Command and Control System Computer Program Project Plan, Volume 1 Development Sperry Univac DSD, June 1973, Sperry Univac DSD.
9. NAVSHIPS 0967-029-6210, Computer Program Performance Specification for DLGN-38 Class Command and Control System Operational Program, Sperry Univac DSD, 15 June 1976, Naval Ships Systems Command.
10. NAVSHIPS 0967-029-6220, Computer Program Design Specification for DLGN-38 Class Command/SIDS), and Control System Operational Program, Sperry Univac DSD, January - September 1976, Naval Ships Systems, January - September 1976, Naval Ships Systems Command.
11. NAVSHIPS 0967-029-6230, Computer Program Performance Specification for DLGN-38 class Combat System Simulation Program, Sperry Univac DSD, January 1973, Naval Ships Systems Command.
12. NAVSHIPS 0967-027-5670, Computer Program operators manual for DLGN-38 Class Combat System Simulation Program. Sperry Univac DSD, Part I March 1975, Part II September 1975, Naval Ships Systems Command.
13. NAVSEA 0967-LP-029-6240, Computer Program Design Specification for DLGN-38 Class Combat System Simulation Program, Sperry Univac DSD, 30 July 1976, Navy, Naval Sea Systems Commands.

DLGN-38 (continued)

14. NAVSHIPS 0967-029-6271, Computer Program Performance Specification for DLGN-38 Class Sensor Interface Data System Program, Sperry Univac DSD, 1 August 1974, Naval Sea Systems Command.
15. NAVSHIPS 0967-029-6280, Computer Program Design Specification for DLGN-38 Class Sensor Interface Data System Program, Sperry Univac DSD, 1 August 1934, Naval Sea Systems Command.
16. NAVSHIPS 0967-029-5720, Interface Design Specification for DLGN-38 Class Command and Control System Operational Program, Sperry Univac DSD, June 1975, Naval Sea Systems Command.
17. NAVSHIPS, 0967-027-5760, Program Certification Document for DLGN-38 class Command and Control System Program, Sperry Univac DSD, Part 1 Certification Plan July 1974, Naval Ships Systems Command.
18. NAVSHIPS, 0967-027-5950, Users Guideline, for DLGN-38 Class Common Program (C&CS/SIDS), Navy, 2 April 1973, Naval Ships Systems Command.
19. NAVSHIPS, CGN-38 Command and Control System (C&CS) Management Plan, NAVSHIPS, April 1976, NAVSHIPS. (see also change 1 dated June 1976).
20. DLGN-38 Command and Control System Computer Program Project Plan Volume 2 Configuration Management and Quality Assurance, Sperry Univac DSD, June 1973, Sperry Univac DSD.

TRIDENT

1. NAVSHIPS 0967-573-0010, TRIDENT Command Program/TRIDENT Service Program, User's Reference Manual, Revision A, 31 March 1975.
2. NAVSHIPS 0967-076-5010, TRIDENT Command and Control System Software Standards and Conventions, Revision A, 1 February 1975.
3. UNIVAC PX 10932, TRIDENT Common Program/TRIDENT Service Program Computer Program Operator's Manual, 12 March 1975, Sperry Univac Navy Systems Division.
4. NAVSHIPS 0900-074-4010, TRIDENT CCS E & I Command and Control Systems Specification (Confidential), 22 April 1974, Naval Ships Systems Command.
5. IBM ICS-TWB-3339, IBM Request for Proposal for CCS TRIDENT Service Program, 13 August 13, 1973, IBM DSD.
6. Statement of Work for the TRIDENT Service Program, 1 November 1973, IBM DSD.
7. UNIVAC PX 10424, Proposal for TRIDENT Service Program, Volume 1, Technical/Management, September 1973, Sperry Univac Navy Systems Division.
8. NAVSHIPS 0900-075-4010, TRIDENT Command and Control System Software Management Plan, Sperry Univac DSD 2 January 1974, Naval Ships Systems Command.
9. NAVSHIPS 0900-074-3010, TRIDENT Submarine CCS Configuration Management Plan, 25 June 1973, Naval Ships Systems Command.
10. IBM Report No. 73-584-012, Data Management Plan, IBM DSD 8 October 1973, IBM DSD.

TRIDENT (continued)

11. NAVSHIPS 0900-075-70X0, CCS E&I TRIDENT Submarine General Computer Complex Design Data Documents, where X represent volume no:

- 1, Operational Narrative
- 2, Hardware Interfaces
- 3, Functional Flow and Operational Sequence Diagrams
- 4, Software Design and Interface Description
- 4, System Design Report
- 4, Function Operational Design

Naval Ships Systems Command.

- 12. UNIVAC PX 11532, Software Quality Assurance Plan for TRIDENT Common Program/TRIDENT Service Program (Revision 6), 1 April 1976, Sperry Univac Navy Systems Division.
- 13. UNIVAC PX 10542, Configuration Management Plan and Procedures for TRIDENT Service Program, 4 January 1974, Sperry Univac Navy Systems Division, File #0059.
- 14. IBM Report No. 75-NJ8-010A, CCS E&I TRIDENT Submarine Reconfiguration Routines Requirements Report, Revision A, 18 October 1975, IBM DSD.
- 15. IBM Report No. 074-L77-002, Revision E, CCS E&I TRIDENT Submarine Degraded Mode Report, 15 September 1975, IBM DSD.
- 16. UNIVAC PX 10540, Computer Program Performance Specification for TRIDENT Service Program, 28 April 1975, Sperry Univac Navy Systems Division.
- 17. UNIVAC PX 10677, Computer Program Design Specification for TRIDENT Service Program, Revision B, 3 August 1975, Sperry Univac Navy Systems Division.

TRIDENT (continued)

18. UNIVAC PX 10731, Computer Program Performance Specification for TRIDENT Common Program, 28 April 1975, Sperry Univac Navy System Division.
19. UNIVAC PX 10840, Computer Program Design Specification for TRIDENT Common Program, Revision B, 27 August 1975, Sperry Univac Navy Systems Division.
20. IBM Report No. 76-NJ8-CC1, CCS E&I TRIDENT Submarine Computer Program Plan for TRIDENT Service Program/TRIDENT Common Program 12 January 1976, IBM DSD.
21. IBM Report No. 76-BJ8-CC4, CCS E&I TRIDENT Submarine Factory Acceptance Test (FAT) for TRIDENT Service Program/TRIDENT Common Program, Revision 6, 15 March 1976, IBM DSD.
22. UNIVAC PX 10651, Quality Assurance Plan for TRIDENT Common Program, 1 March 1974, Sperry Univac Navy Systems Division.
23. UNIVAC PX 10545, Quality Assurance Plan for TRIDENT Service Program, 1 April 1974, Sperry Univac Navy Systems Division.
24. IBM 74-TRI-0023, ENG-ELB-0416, TRIDENT Command and Control System Test and Evaluation Management Plan, Naval Ships Systems Command (PMS-396), November 1973, Naval Ships Systems Command.
25. UNIVAC PX 10523, Management and Performance Plan for TRIDENT Service Program, Sperry Univac DSD.
26. UNIVAC PX 11608, Computer Program Test Procedures for Revision 6. TRIDENT Common Program/TRIDENT Service Program, Sperry Univac DSD, 9 June 1976, Sperry Univac DSD.
27. IBM Report No. 76-NJ8-001, CCS E&I TRIDENT Submarine Computer Program Test Plan for Release 6 TRIDENT Common Program/TRIDENT Service Program, IBM DSD, 12 January 1976, IBM DSD.

TRIDENT (continued)

28. IBM Report No. 74-NJ8-007, CCS E&I. TRIDENT Submarine TRIDENT Service Program Design Review Comments IBM FSD 18 November 1974, General Dynamics Electric Boat Division.
29. UNIVAC PX 11533, Software Configuration Management Plan and Procedures for TRIDENT Common Program/TRIDENT Service Program (Revision 6), Sperry Univac DSD, 1 April 1976, Sperry Univac DSD.
30. Sperry Univac PX 10424 TRIDENT Service Program (TSP) Volume 1. Technical/Management Proposal, Sperry Univac, Sept. 1973, Sperry Univac DSD.
31. IBM 73-TRI-0569, Command and Control System Engineering and Integration (CCS E&I) TRIDENT Submarine Data Management Plan IBM, 8 Oct. 1973, IBM.

TRIDENT SHIP CONTROL

1. NAVSHIPS 0900-076-9010 TRIDENT Software Management and Performance Plan for the TRIDENT Ship Control Systems, Sperry Univac DSD, May 1974, Naval Ships Systems Command.
2. UNIVAC File No. 0062, Statement of Work for TRIDENT Ship Control Program, Electric Boat, 14 January 1974, Sperry Univac Navy Systems Division.
3. NAVSHIPS 0900-076-6010, Computer Program Performance Specification for TRIDENT Ship Control Systems, Electric Boat, 1 April 1974 Naval Ships Systems Command.
4. NAVSEA 0967-LP-080-5010A, Computer Program Test Plan for TRIDENT Ship Control Systems Electric Boat, 11 February 1976, Naval Sea Systems Command.
5. NAVSEA 0967-LP-080-7010, Computer Program Test Procedure for TRIDENT Ship Control Application Program, Sperry Univac DSD, July 1976, Naval Sea Systems Command.
6. NAVSEA 0900-080-6020, TRIDENT Command and Control System Simulation and Test Program Operators and Users Manual for Ship Control Systems, Sperry Univac DSD, October 1975, Sperry Univac DSD.
7. NAVSHIPS 0900-078-2010, Computer Program Design Specification for TRIDENT Ship Control System, Sperry Univac DSD, 13 September 1974, Naval Ships Systems Command.
8. NAVSHIPS 0900-078-3010, Computer Program Performance Specification for TRIDENT Ship Control Simulation Program, Electric Boat, 3 December 1975, Naval Ships Systems Command.
9. NAVSEA 0900-LP-0800-1010, Computer Program Design Specification for TRIDENT Ship Control Simulation Program, Sperry Univac DSD, 3 December 1975, Naval Ships Systems Command.

TRIDENT SHIP CONTROL (continued)

10. Informal Documents, TRIDENT Ship Control Trainer AN/UYK-7 Software Package Performance and Design Description, Sperry Univac DSD, 25 June 1976, Sperry Univac DSD.
11. Statement of work for TRIDENT Ship Control Program, Sperry Univac DSD, 14 Jan 1974, Sperry Univac Naval Systems File #0062.

Common Program

1. NAVSHIPS 0967-027-5160, Computer Program Operators Manual for AN/UYK-7 Simulator, 15 January 1975.
2. Contract No. N00024-69-C-1233, Univac Technical Note 301A, Guidelines for Common Program MODX1, 1 June 1971, Naval Ship Systems Command.
3. NAVSHIPS 096-7-011-0160, Configuration Management Plan for Standard Executive Program Module (MODX2), April 1974, Sperry Univac, Naval Ships Systems Command.

Other Univac

1. Remote Integrated Programming System User's Guide, August 1974, Sperry Univac Navy System Division.
2. Remote Integrated Programming System User's Guide Update 1, April 1976, Sperry Univac Navy Systems Division.
3. UP-4144.3, UNIVAC 1100 Series Executive System Programmer Reference, Sperry Univac.
4. UNIVAC Manual 4000 series, Standard Practices Manual for Engineering and Production Operations.
5. Sperry Univac PX-7782, Flow Charts (flow chart plotted with CHAR actors and template symbols), Sperry Univac DSD.
6. Sperry Univac Document No. 1117, Graph Users Guide, Sperry Univac DSD, 15 May 1975, Sperry Univac DSD.
7. Sperry Univac (informal document), Mapper General Data Policy, Sperry Univac DSD, Data prior to 27 May 1975, Sperry Univac DSD.
8. Sperry Univac (Informal Document), Management Responsibility reporting (MRR), Sperry Univac DSD, Date prior to October 1975, Sperry Univac DSD.
9. Sperry Univac PX-11434, Computer Aided Techniques Documentation System - Design Object Drawing Outputter "DODO" (Level 1.3) User's Guide, Sperry Univac DSD, 14 Nov. 1975 Sperry Univac DSD.
10. Sperry Univac PX 11451, SSN 688/700 Processing Evaluator Tool (PET MODULE) Operating Description, Sperry Univac September 1975, Sperry Univac DSD.
11. Sperry Univac PX 11243, PROMIS Program Management Information System User's Manual Volume 1 PROMIS Guidelines, Volume 2 Appendixes, Sperry Univac, Revised 28 July 1975, Sperry Univac DSD.

Other Univac (continued)

12. Sperry Univac (no number) Remote Integrated Programming System (RIPS) User's Guide Sperry Univac DSD, August 1974.
Sperry Univac DSD Also User's Guide Updated 1 April 1976.
13. Sperry Univac (Informal Document), Why Structured Narratives, Sperry Univac, no date, Sperry Univac DSD.

Other Navy

1. WS-8506 Revision 1, Requirements for Digital Computer Program Documentation, 1 November 1971, Naval Ordnance Systems Command.
2. NAVSHIPS 0967-028-0060, User's Reference Manual for Compiler- Monitor System (CMS-2) for use with AN/UYK-7 Computer, Naval Systems Command. October 1973.
3. NAVSEA 0967-LP-024-5452, Computer Set AN/UYK-7 (V) Diagnostic Program Operating Procedures, 1975, Naval Sea Systems Command.
4. UNIVAC PX 5652E, AN/UYK-7 Technical Description, Sperry Univac, DSD.
5. NAVSHIPS 0967-051-6291, Equipment Specification for Computer Set An/UYK-7(V), 1 November 1973, Naval Ships Systems Command.
6. NAVSHIPS 0967-011-0011, Specifications for Digital Computer Program Documentation, US Navy, 1 October 1968, Naval Ships Systems Command.

METRIC SYSTEM

BASE UNITS:

Quantity	Unit	SI Symbol	Formula
length	metre	m	...
mass	kilogram	kg	...
time	second	s	...
electric current	ampere	A	...
thermodynamic temperature	kelvin	K	...
amount of substance	mole	mol	...
luminous intensity	candela	cd	...

SUPPLEMENTARY UNITS:

plane angle	radian	rad	...
solid angle	steradian	sr	...

DERIVED UNITS:

Acceleration	metre per second squared	...	m/s
activity (of a radioactive source)	disintegration per second	...	(disintegration)/s
angular acceleration	radian per second squared	...	rad/s
angular velocity	radian per second	...	rad/s
area	square metre	...	m
density	kilogram per cubic metre	...	kg/m
electric capacitance	farad	F	A·s/V
electrical conductance	siemens	S	A/V
electric field strength	volt per metre	...	V/m
electric inductance	henry	H	V·s/A
electric potential difference	volt	V	W/A
electric resistance	ohm	...	V/A
electromotive force	volt	V	W/A
energy	joule	J	N·m
entropy	joule per kelvin	...	J/K
force	newton	N	kg·m/s
frequency	hertz	Hz	(cycle)/s
illuminance	lux	lx	lm/m
luminance	candela per square metre	...	cd/m
luminous flux	lumen	lm	cd·sr
magnetic field strength	ampere per metre	...	A/m
magnetic flux	weber	Wb	V·s
magnetic flux density	tesla	T	Wb/m
magnetomotive force	ampere	A	...
power	watt	W	J/s
pressure	pascal	Pa	N/m
quantity of electricity	coulomb	C	A·s
quantity of heat	joule	J	N·m
radiant intensity	watt per steradian	...	W/sr
specific heat	joule per kilogram-kelvin	...	J/kg·K
stress	pascal	Pa	N/m
thermal conductivity	watt per metre-kelvin	...	W/m·K
velocity	metre per second	...	m/s
viscosity, dynamic	pascal-second	...	Pa·s
viscosity, kinematic	square metre per second	...	m/s
voltage	volt	V	W/A
volume	cubic metre	...	m
wavenumber	reciprocal metre	...	(wave)/m
work	joule	J	N·m

SI PREFIXES:

Multiplication Factors	Prefix	SI Symbol
1 000 000 000 000 = 10 ¹²	tera	T
1 000 000 000 = 10 ⁹	giga	G
1 000 000 = 10 ⁶	mega	M
1 000 = 10 ³	kilo	k
100 = 10 ²	hecto*	h
10 = 10 ¹	deka*	da
0.1 = 10 ⁻¹	deci*	d
0.01 = 10 ⁻²	centi*	c
0.001 = 10 ⁻³	milli	m
0.000 001 = 10 ⁻⁶	micro	μ
0.000 000 001 = 10 ⁻⁹	nano	n
0.000 000 000 001 = 10 ⁻¹²	pico	p
0.000 000 000 000 001 = 10 ⁻¹⁵	femto	f
0.000 000 000 000 000 001 = 10 ⁻¹⁸	atto	a

* To be avoided where possible.